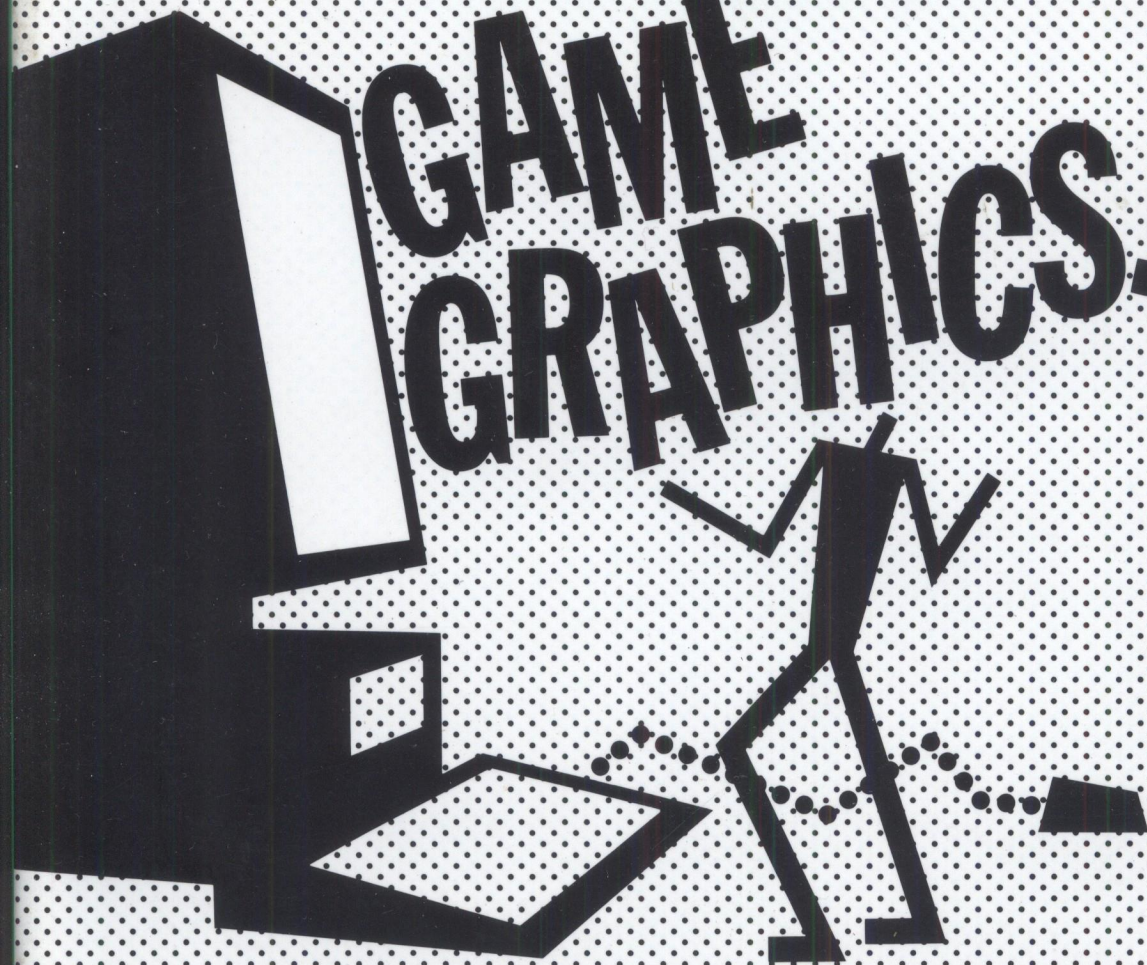
POPCOM BOOKS

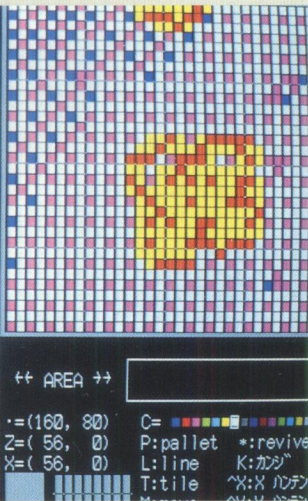
5インチ2HDディスク付

PC-9801シリーズ

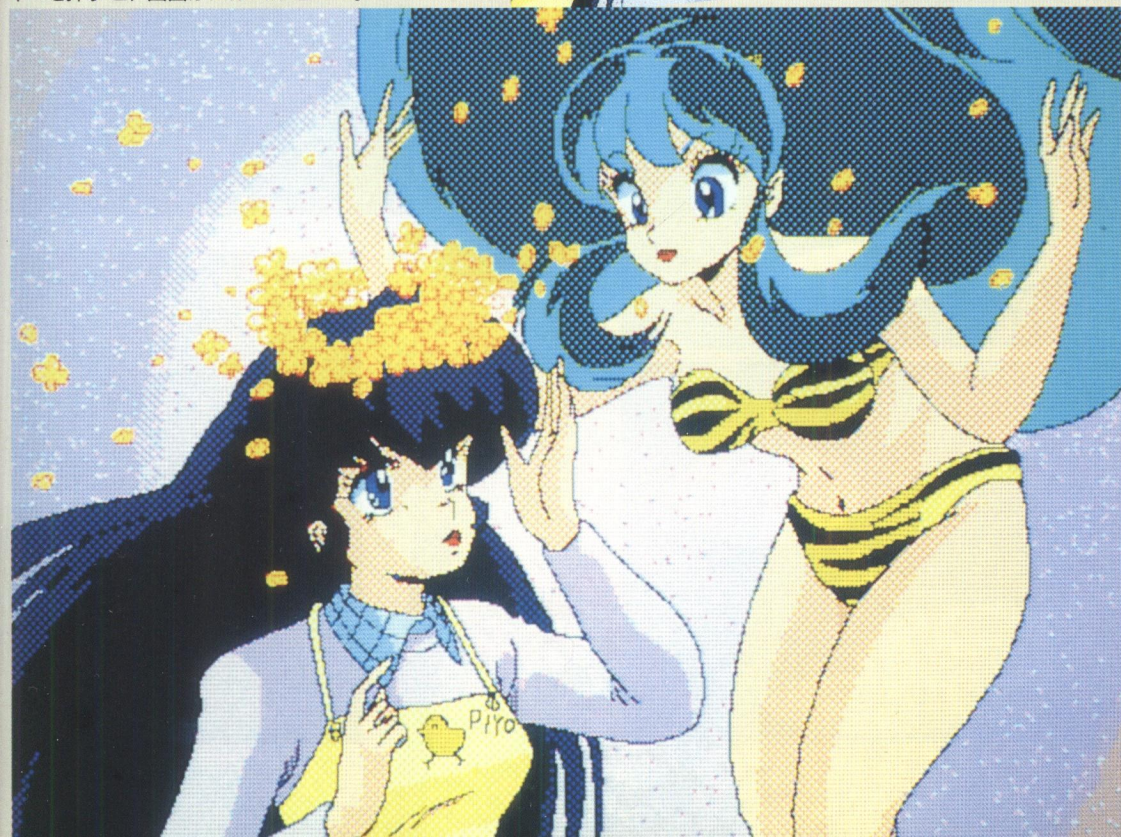
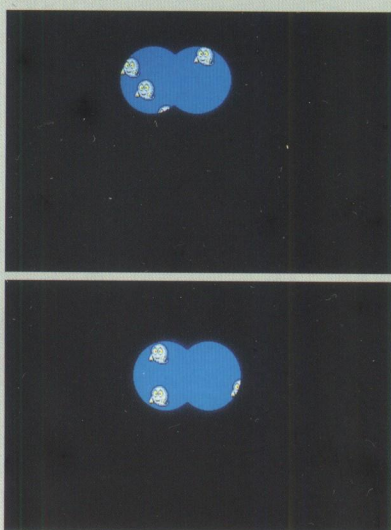
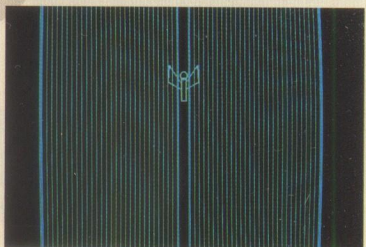
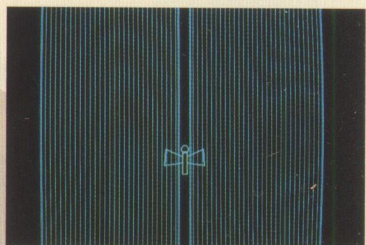
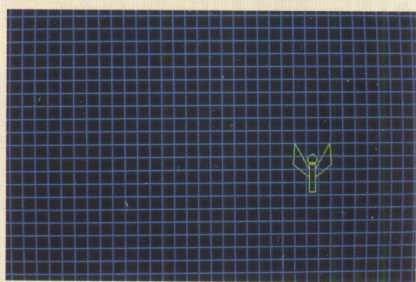
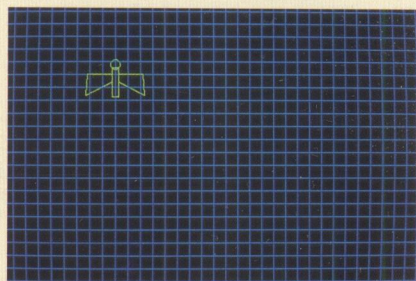
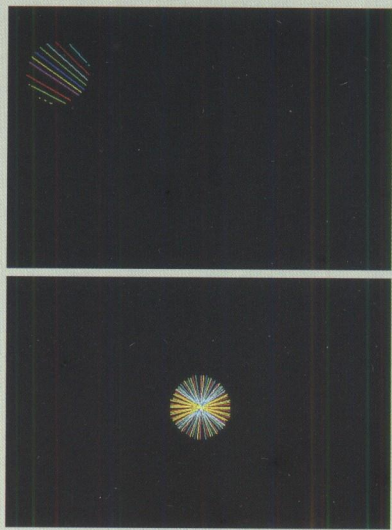
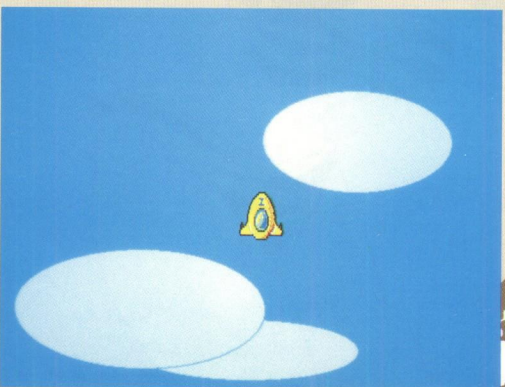
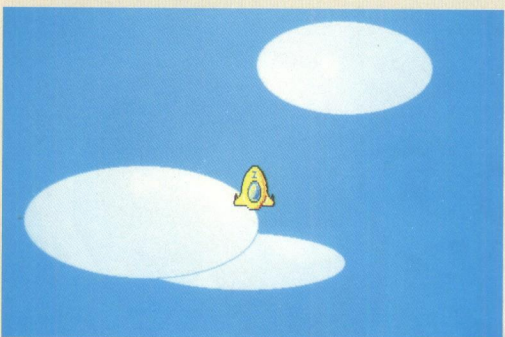
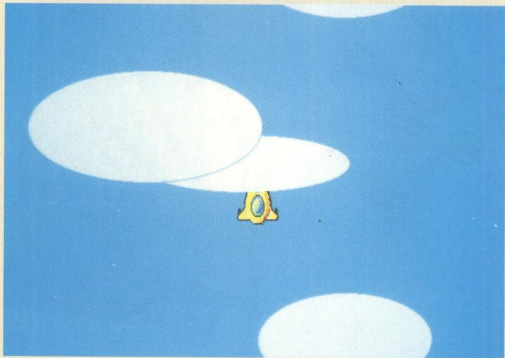
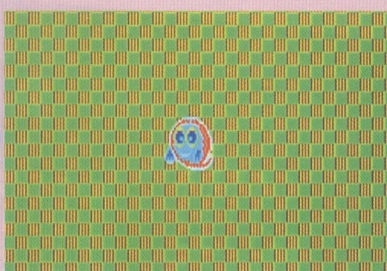
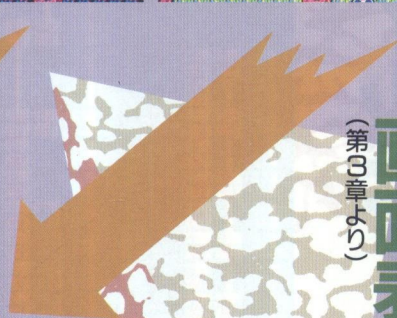
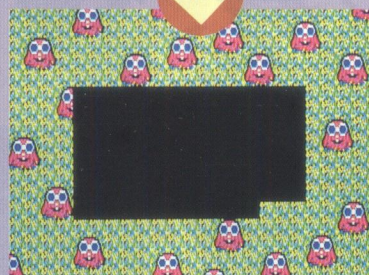
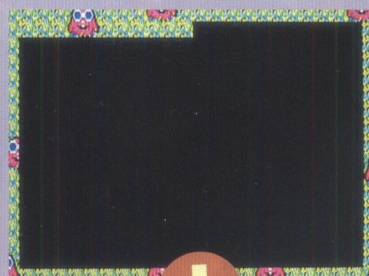
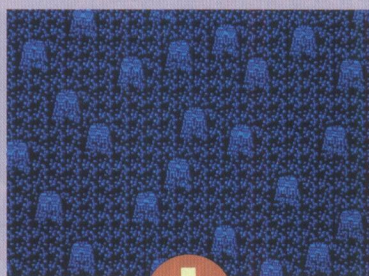
マシン語ゲームグラフィックス

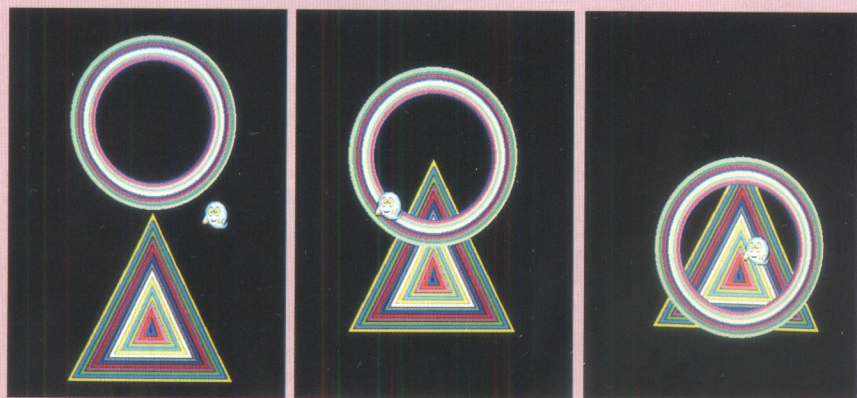
日高 徹/青山 学・共著

小学館

ワ！大型グラフィックスができる！

本文第5章「魅惑の大型グラフィックス」より

●グラフィック・ツール『PMAN 98』により、フル画面のエディットが自由自在！
●原画の入力は、白黒／カラースキャナ対応の『SCAN 98』でラクラク！

●エディットした絵は、圧縮をかけてセーブできるので、ゲームへの応用も！

●アララ、画面がスクロールした……なんて驚かないで！　エディット中に←→キーを押すと、画面がスルスルと……。

パレット操作により不定形マスクが（第3章より）

テキスト・マスクで気らーくにスポット処理（第3章より）

パレット操作によるプレーン分割のいろいろ（第3章より）

おばけのピーヨがチカチカと反転
（第2章より）
おしゃれな画面表示・消去法
（第3章より）
ジワジワ表示・消去、渦巻型の表示・消去、斜め表示・消去と、工夫しだいでいろいろできる！

これが多重スクロールだ！

いまや、ゲームに必須の多重スクロール。最新のテクニックを大公開！　わかりやすい解説で、驚異のハイテクもバッチリ身につくぞ！（第4章より）

重ね合わせアニメーション処理とは？

テキスト画面マスクと4面同時転送によるアニメーションテクニック（第3章より）

PC-9801シリーズ

マシン語ゲームグラフィックス

小　学　館

はじめに

コンピュータの世界では、マシン語以外の言語（BASIC や FORTRANなど）を「マシン語に比べれば人間の言葉に近い」ということで高級言語と称していますが、実際の人間の言葉というものは常に変化し進化するものです。例えば流行語などはその典型で、次から次へと生まれてははかなく消えていきます。

このように流動的な人間の言語に比べ、コンピュータにおける高級言語は変化や進化とはほとんど無縁のものです。もちろん、機種による違いや部分的に拡張されることはありますが、ユーザーが独自に言語を発展させることは不可能な状況です。つまり、BASIC で書かれたプログラムは5年前のものでも古さを感じることはなく、いわゆる文語体／口語体のような区別をつけるのは困難です。

一方、マシン語というのは単体ではアルファベット的なものですから、組み合わせて初めて言語としての役割を果たすものです。一見すると原始的に思えますが、人間の言葉もルーツをたどればマシン語以上に原始的です。だからこそ、変化し発展することができるわけです。マシン語が言葉になるとき、人はそれをテクニックと言います。自分自身の技術的成長もありますが、こういったテクニックというものは時代とともに変化し進歩していくものです。そこには明らかに流行もあるし、過去形となった旧式のテクニックもあります。こう考えると、まさにマシン語こそが真の高級言語なのかもしれません。

そして、こういった流行に最も敏感に反応するのがゲームにおけるテクニックです。中でも、グラフィックスはゲームの顔であり象徴ですから、常に最新の技術が求められており、またプログラマーの腕の見せどころでもあります。

これまで、多くの読者の方々から、様々な要望や激励のお便りを戴いていますが、今回は特に希望の多い最新のグラフィックス・テクニックを集め、できるだけわかりやすく解説したつもりです。ゲームそのものの謎解きも楽しいでしょうが、ゲームに隠されたマシン語の謎について考えるのは、一段上の知的ゲームです。本書が、そのマシン語の謎を解決する一助になれば筆者としてうれしい限りです。

また、グラフィックスを語る上において、グラフィック・ツールの存在を無視することはできません。本書では、グラフィックス開発用支援ツールとして、大型グラフィックス編集用に『PMAN98』、スキャナコントロール用に『SCAN98』、キャラクタ・パターン用に『PTER98』という3つのツールを用意しました。これらは私がプロ用に開発したバージョンを一般用に使いやすくリメイクしたもの

です。市販商品のように実用から離れたハデな部分はカットしましたが、ゲームを作る上においてはいずれも十分な実力を備えているものばかりです。きっと、あなたの PC-9801に BASICでは味わえなかった楽しい世界が広がることでしょう。

なお、本書執筆にあたり日本電気株式会社、オムロン株式会社よりデモ機材等の貸し出しを受けましたことを、この場より改めて御礼申し上げる次第です。

1991年3月1日　著者

＝＝＝　目　次　＝＝＝

《3.5インチ2HD版サービスについて》

3.5インチ版対応の機種をお持ちの方のために、実費（送料込み500円）でサービスいたします。セーブされている内容は、5インチ版と全く同じです。

★申し込み方法★

まず、郵便局で500円の**『定額小為替証書』**をお買い求めください。
つぎに、便せんにあなたの**住所・氏名・電話番号**を明記して、小為替証書を同封し、封書にて下記あてお送りください。
なお証書には受取人名などを書き込まないようにご注意ください。

〈申し込み先〉

〒101　東京都千代田区神田神保町3-3-7
　　　昭和第2ビル

（株）新企画社

98マシン語ゲームグラフィックス3.5インチ版サービス係

・商品の到着は申し込み受付け後、2週間ほどかかります。

第1章　グラフィックスの基礎

コンピュータと人間を結ぶ最大のインターフェース、それはディスプレイ装置（標準的にはブラウン管）による画面表示です。もしも、コンピュータが画面に何も出力してくれなかったなら、われわれはコンピュータをどのように利用していいか……。プログラムを組むことはおろか、ゲームで遊ぶこともできずに右往左往するばかりでしょう。

普段、無意識のうちに操作していることのすべてが、実はディスプレイに表示するというプログラムのお陰で成立していたのです。つまり、画面表示はあらゆるプログラムの窓であり、けっして避けては通れない基礎テクニックなのです。本当の主役は画面に現れないプログラムでも、第三者にとっての主役は常に画面です。まさに「画面を制する者がすべてを制す」と言っても過言ではないでしょう。

そのための第一歩、それは PC-9801シリーズにおける画面表示の能力を正確に把握することです。市販ゲームの高度な画面処理も、すべてはその範囲の中でテクニックを駆使した結果なのですから……。

1－1　表示画面の種類

初代の PC-9801が発売されたのは1982年のことでした。その後、幾度もモデルチェンジが重ねられましたが、画面表示の基本部分に関しては一貫した構造がとられています。もちろん、表示処理速度のアップやプレーンの追加、カラーパレットのアナログ化（4096 色中から16色選択可能）など、細かな改良点はアチコチにあります。しかし、画面を構成しているVRAM（ビデオラム）* の構造は、同じように設計されているのです。

すでにご存じの方も多いと思いますが、とりあえず PC-9801シリーズにおける画面構成を確認することにしましょう。図 1-1-1 を見てください。

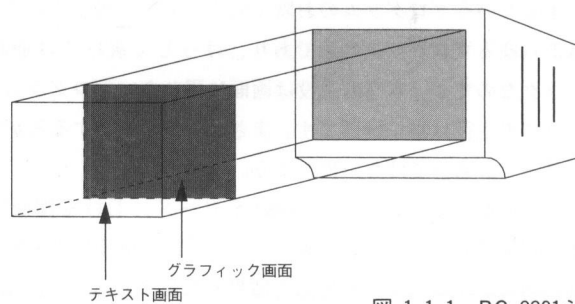



図 1-1-1　PC-9801シリーズの画面構成

この図で示したように、PC-9801 にはテキスト画面とグラフィック画面とがあり、テキスト画面のほうが表示優先順位が高いことがわかります。つまり、グラフィック画面に絵が表示されていても、テキスト画面に文字が表示されれば、文字の部分だけ絵が隠れてしまうということです。また、グラフィック画面にはカラーモードと白黒モードがあり、BASIC では次のように使い分けることができます。

★カラーモード：

640×200 ドットで４画面（画面モード 0）

640×400 ドットで２画面（画面モード 3）

☆白黒モード：

640×200 ドットで16画面（画面モード 1）

640×400 ドットで８画面（画面モード 2）

PC-9801 にはグラフィック画面として４枚のプレーン*(旧タイプでは３枚）が第

VRAM（ビデオラム）：

その中身が直接画面表示に関係する特殊なメモリ。画面に文字を出したり、絵を表示したり……と、画面はこのビデオラムに入れられたデータによって創られている。

４枚のプレーン：

正しくはグラフィック・ビデオラム（G.VRAM）０～３と呼ばれるが、本書ではカラーモード／白黒モードを問わずＢ面／Ｒ面／Ｇ面／Ｉ面という表記で区別をしている。これは、カラーモードでこのほうがわかりやすいので、白黒モードでも統一したほうがプレーンを一致させやすいからである。

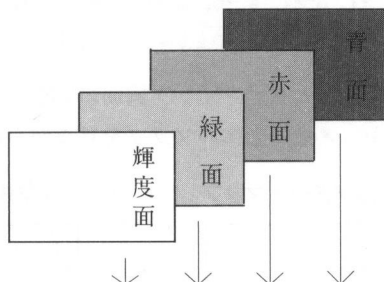
1面用と第2画面用の2組（初代PC-9801、PC-9801U等では第1画面のみ）用意されており、実際にはこの第1と第2を合わせて8枚のプレーンをモードによって使い分けています。

第1画面と第2画面は互いに独立しており、メモリ・マップの同じ物理アドレス上に配置されていて、ポートA4，A6で次のように使い分けしています(図 1-1-2)。

図 1-1-2

値＼ポートNo.	0A4H	0A6H
0	第1画面表示	第1画面アクセス
1	第2画面表示	第2画面アクセス

これらの画面表示と画面アクセスの組み合わせは任意に取ることができます。なお、本書では主にこの第1画面に対して記述していきますが、どちらの画面でも機能的にはまったく変わりません。

さて、標準的なカラーモードでは、第1画面の4枚のプレーンをB（Blue＝青）面、R（Red＝赤）面、G（Green＝緑）面、I（Intensity＝輝度）面とに分け、それを組み合わせることでカラー16色表示を実現しています（図 1-1-3）。

図1-1-3　グラフィック画面の
カラー発色の仕組み

I 面	G 面	R 面	B 面	カラー番号
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8
1	0	0	1	9
1	0	1	0	10
1	0	1	1	11
1	1	0	0	12
1	1	0	1	13
1	1	1	0	14
1	1	1	1	15

0:そのプレーンが発色しない
1:そのプレーンが発色する

これに対し、白黒8面モードというのはこの第1と第2合わせて8枚のプレーンを、それぞれ独立したプレーンとして使うモードです。

これは、各プレーンを個別にアクセスできるマシン語では、パレットによっても同様の働きをさせることが可能となっています。例えば、B面の内容を見たければ、B面の関わっているカラー＝1（他のカラーでもよい）とし、B面の関わっていないカラー＝0とすればいいのです。

パレット指定：

0→0, 1→1, 2→0, 3→1, 4→0, 5→1, 6→0, 7→1
8→0, 9→1,10→0,11→1,12→0,13→1,14→0,15→1

この方法であれば、特定のプレーンを好みのカラーで見られるばかりでなく、面に優先順位をつけて合成するということも可能です。

（例）B面（カラー7で表示）とR面（カラー1で表示）を合成する。表示はB面を優先する(合成してダブる部分はカラー7となる)。

パレット指定：

0→0, 1→7, 2→1, 3→7, 4→0, 5→7, 6→1, 7→7
8→0, 9→7,10→1,11→7,12→0,13→7,14→1,15→7

では、次にテキスト画面とグラフィック画面のVRAM(ビデオラム)がメモリのどこに存在しているのか、メモリ全体のマップを見ながら確認することにしましょう。

・基本的なメモリ・マップ

アドレス	内容
FFFFF–E8000	BASIC ROM エリア
E8000–E0000	グラフィック VRAM 32KB×2
E0000–C8000	
C8000–C0000	ユーザー ROM エリア
C0000–A8000	グラフィック VRAM 96KB×2
A8000–A5000	
A5000–A4000	CG ウインドウ
A4000–A0000	テキスト VRAM 12KB
A0000–00400	メイン RAM
00400–00000	割り込みベクトル

・VRAMの構成

アドレス	内容
E7FFF–E0000	グラフィック VRAM 輝度面
BFFFF–B8000	グラフィック VRAM 緑色面
B8000–B0000	グラフィック VRAM 赤色面
B0000–A8000	グラフィック VRAM 青色面
A3FFF–A2000	アトリビュート・テキスト VRAM:偶数番地
A2000–A0000	テキスト VRAM

図 1-1-4 PC-9801の基本的なメモリ・マップとVRAMの構成

図 1-1-4から、テキストVRAMは物理アドレスで A0000H番地から12KBを割り付けてあることがわかります。また、グラフィックVRAM（以下 G.VRAM）は A8000Hから96KB×2と、E0000Hから32KB×2（PC-9801／E／F／Mでは実装不可）となります。

なお、G.VRAMの表示域は、最大で各セグメントのオフセット・アドレス 0〜7CFFHの範囲となります。7D00H〜7FFFHは実際には画面上に表示されません。また、表示画面としては、0〜7CFFH全体を1ページとする640×400モードと、G.VRAMの上半分の0〜3E7FHと下半分の3E80H〜7CFFHを、それぞれ1ページとして使用する640×200モードとがあります。これらのモードはG.VRAMをアクセスする前に、あらかじめ設定しておきます。

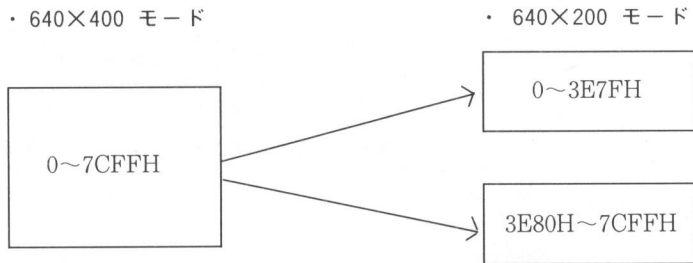
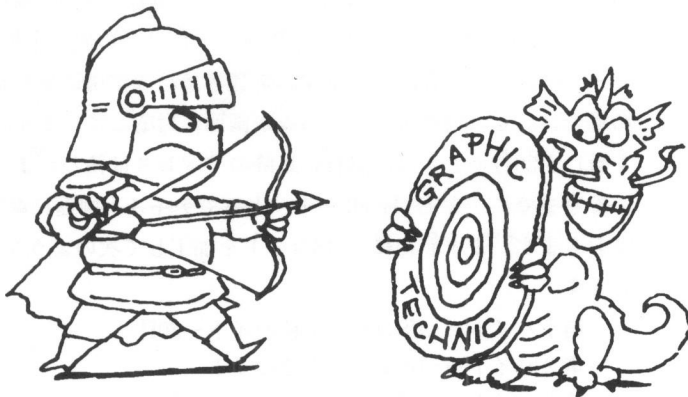



図 1-1-5　表示モードの設定

1－2　画面操作のコモンセンス

すでにマシン語でプログラムを組まれている方にとって、この程度の内容は知っていることばかりと感じるかもしれません。しかし、市販されているゲームの高度なテクニックも、その知っていることを応用しているに過ぎないのです。基礎を振り返ることで、新しい何かを発見することが往々にしてあります。知識をリフレッシュするつもりで、テキストVRAMとグラフィックVRAMの初歩的な使い方を確認しておくことにしましょう。

テキストVRAMは図 1-1-4で示されていたように、メインメモリのA000:0000H～A000:3FFFHを占めています。これをもう少し詳しくしたのが図 1-2-1です。

ページ	文字コードエリアCPUアドレス	属性エリアCPUアドレス(偶数番地のみ)
1	A000:0000H ～ A000:0FFFH	A200:0000H ～ A300:0FFFH
2	A100:0000H ～ A100:0FFFH	A300:0000H ～ A300:0FFFH

第1ページ　　　　第2ページ

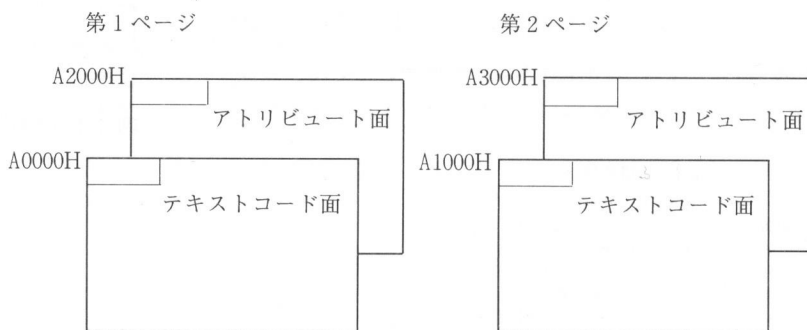



図 1-2-1　テキストVRAMの構造

テキストVRAMは、図のように2ページ用意されていますが、2ページ目は未使用となっています。また、1つの文字に対して、文字コード用と文字に対する属性用の2つのVRAMがありますが、テキスト画面の桁数(横の文字数)が80(WIDTH 80)に設定されている場合、文字コードエリアの偶数番地の1バイトは画面のアスキーコード1文字に対応します。奇数番地は漢字用に使われるものです。桁数が40の場合は、4の倍数アドレスのみが画面に対応しています。なお、行数(縦のライン数)が20行のときは、21行目以降のアドレスは使われません。

参考までに、WIDTH 80のモード下でテキスト画面に漢字を表示する手段を示しておきます。まず最初に、JISコードを加工して次のような4バイトの数値を作り出します。

第1バイト・・・JISコード上位バイト－20H
第2バイト・・・JISコード下位バイト
第3バイト・・・JISコード上位バイト－20H

第４バイト・・・JISコード下位バイト＋80H

次に、この第１～第４バイトの順で、連続した文字コードエリア（奇数番地も含む）へ書き込みます。具体的に漢字『渦』で考えてみましょう。『渦』のJISコードは3132Hですから、先ほどの４バイトの数値は次のようになります。

第１バイト・・・31H－20H＝11H
第２バイト・・・　　　　　　32H
第３バイト・・・31H－20H＝11H
第４バイト・・・32H＋80H＝B2H

これを第１バイトから順に文字コードエリアの連続したアドレスへ書き込むわけです。アトリビュート（属性）エリアについては、テキスト画面の１文字に対してカラーやブリンクといった属性を付けるためのもので、偶数番地のみが使われます。奇数番地のメモリは存在していません。

b7	b6	b5	b4	b3	b2	b1	b0
緑	赤	青	VL BG	UL	RV	BL	ST

ST: シークレット	b0＝0で有効	b0＝1でノーマル表示
BL: ブリンク	b1＝1で有効	b1＝0でノーマル表示
RV: リバース	b2＝1で有効	b2＝0でノーマル表示
UL: アンダーライン	b3＝1で有効	b3＝0でノーマル表示

(b4＝1) & (ポート68H: Mode F／F ADR＝0 & DT＝0)であれば
VL: バーチカルライン表示

(b4＝1) & (ポート68H: Mode F／F ADR＝0 & DT＝1)であれば
BG: 簡易グラフパターン表示

b4＝0 であればノーマル表示

b5	1	0	1	0	1	0	1	0	
b6	1	1	0	0	1	1	0	0	
b7	1	1	1	1	0	0	0	0	
	白色	黄色	水色	緑色	紫色	赤色	青色	黒色	カラーCRT
	明 ← 中 → 暗								モノクロCRT 濃淡表示

図 1-2-2　アトリビュートの内容

属性の内容は、白黒モードとカラーモードでは**図 1-2-2** に示されているように同じではありません。また、b4＝1とした時は、ポート68Hを介してVL／BGをコントロールします。0を出力すれば垂線表示（VL）であり、1を出力すれば簡易グラフィック表示（BG）となります。

ここで属性のところにある簡易グラフィックについて少々説明を加えておきます。これは正確にはテキスト画面によるロー・レゾリューション・グラフィック（160×100 ドット）のことで、その昔PC-8801の前身であるPC-8001（併売されていた時期もある）において使用されていたN-BASICモードのグラフィックスと同じものです。640×200 ドットのフルカラー・グラフィックと比べてもオモチャのような存在ですが、グラフィック画面と簡単に重ね合わせることができるので、いろいろな用途があります。具体的な活用例はこの先折りに触れ紹介しますので、ここではテキスト・グラフィックの存在とビット別の内容を覚えておいてください。

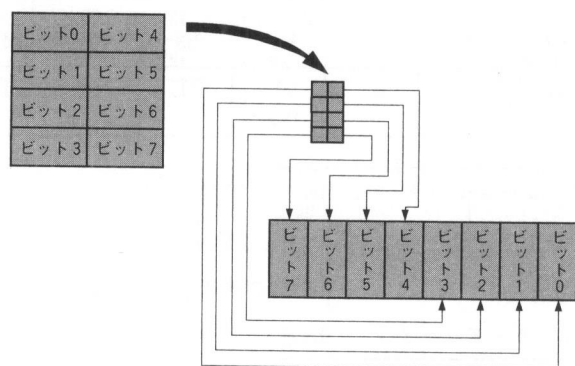



図 1-2-3　テキスト・グラフィックのビット別構成

次にグラフィック画面についてですが、MS-DOSではディスクBASICと違ってグラフィックス・システムをユーザーが自由に選択して使えるようになっています。そこで、利用前にグラフィックス・システムの初期化と画面モードの選択をしておかなければなりません。グラフィックス・システムの初期化には、次の3つの方法があります。

① MS-DOSのデバイスドライバにグラフィックスドライバを組み込む。
② PC-9801のROM内ルーチンを利用する。
③ 独自にハードウェアをコントロールする。

もし、ユーザーのシステムが8色中8色のモードを選択する場合ならば、最も簡単な方法はPC-9801内のROM内ルーチンを利用する方法です。しかし、このROM内ルーチンは8色中8色のモードにしか対応していません。そのため、4096色中16色モードを使いたい場合は、MS-DOSのシステムコール（ファンクションコール）を使う方法が最も簡単です。

とはいえ、MS-DOSのグラフィクスドライバを使う為には、それなりの手続きを

取らなければなりません。まず、MS-DOSのグラフィックスドライバ《GRAPH.SYS》を《CONFIG.SYS》に登録します。これは、エディタなどで《CONFIG.SYS》に次のように1行加えます。

```
A>TYPE CONFIG.SYS
    ⋮
DEVICE=GRAPH.SYS            ←この1行を加える
    ⋮
```

次に、MS-DOSシステムを再起動します。もちろん、そこには《GRAPH.SYS》と《GRAPH.LIB》の2つのファイルが存在していなければなりません。これで、グラフィックス・システムを使う為のシステム側の準備が整ったわけですが、今度はグラフィックスドライバを使用可能にするソフトが必要です。それがLIST 1-0です。

LIST 1-0

```
;************************************
;*           LIST1-0              *
;************************************

PUBLIC  GSTAT,GTERM,GDSET,GCALL,GMD16,GMOD8,GM200

CODE    SEGMENT PUBLIC

        ASSUME CS:CODE,DS:CODE

PRINT   MACRO   STRING                      ;;文字列の出力マクロ定義
        MOV     DX,OFFSET STRING
        MOV     AH,09
        INT     21H
        ENDM

G_START    EQU     0                        ;グラフィックスの開始コード
G_TERM     EQU     1                        ;グラフィックスの終了コード
G_MODE_SET EQU     3                        ;表示モードの設定コード
G_SWITCH   EQU     11                       ;表示スイッチの設定コード

GSTAT   PROC                                ;グラフィックスの開始プロシージャ定義
        MOV     AX,CS                       ;AX=CS
        MOV     DS,AX                       ;DS=AX=CS
        CALL    CDRIV                       ;グラフィックスドライバの存在確認
        JB      GSTRT                       ;ドライバがなければGSTRTへ
        MOV     GSTSIN,1                    ;グラフィックスシステムの開始とする
        PUSH    DS                          ;DSのセーブ
        MOV     BX,OFFSET GDADR             ;BX←エントリー格納アドレス
        XOR     AX,AX                       ;AX←0
        INT     0CDH                        ;DS:BXへエントリーテーブルのアドレスを取得
        MOV     SI,G_START                  ;SI←グラフィックス開始コード
        CALL    GCALL                       ;グラフィックスの開始
        CALL    GDSET                       ;DS:SI←パラメータ・ブロックの先頭アドレス
        MOV     AL,0FH                      ;表示スイッチを表示状態とする
        MOV     [SI+4],AL                   ;パラメータ・セット
        MOV     SI,G_SWITCH                 ;SI←ファンクション・コード
        CALL    GCALL                       ;ファンクション・コール
        POP     DS                          ;DS値を復元
        CLC                                 ;正常終了(CY=0)とする
GSTRT:  RET                                 ;リターン
GSTAT   ENDP                                ;プロシージャの終了
```

```
D_NAME  DB      'GRAPH  $',0                ;デバイス名

CDRIV   PROC                                ;ドライバ・チェック用プロシージャ定義
        MOV     AX,3D00H                    ;ドライバ・オープンとする
        MOV     DX,OFFSET D_NAME            ;パス名格納アドレス
        INT     21H                         ;ファンクション・コール
        JC      CDR01                       ;エラーであればCDR01へ
        MOV     BX,AX                       ;BX←ハンドル・セット
        MOV     AH,3EH                      ;ドライバのクローズ
        INT     21H                         ;ファンクション・コール
        CLC                                 ;正常終了(CY=0)とする
        JMP     CDRRT                       ;CDRRTへ
CDR01:  PRINT   GOPER                       ;エラー・メッセージ表示
        STC                                 ;異常終了(CY=1)とする
CDRRT:  RET                                 ;リターン
CDRIV   ENDP                                ;プロシージャの終了

GMD16   PROC                                ;16色モード設定プロシージャ
        MOV     DX,103H
        CALL    GMDST
        RET
GMD16   ENDP

GMOD8   PROC                                ;8色モード設定プロシージャ
        MOV     DX,101H
        CALL    GMDST
        RET
GMOD8   ENDP

GM200   PROC                                ;640×200，8色モード
        MOV     DX,1
        CALL    GMDST
        RET
GM200   ENDP

GMDST   PROC                                ;モード設定用プロシージャ
        PUSH    DS                          ;DS値セーブ
        CALL    GDSET                       ;DS:SI←パラメータ・ブロックの先頭アドレス
        XOR     AX,AX                       ;AX←0
        MOV     [SI],AX                     ;第1パラメータ ← 0
        MOV     [SI+2],AX                   ;第2パラメータ ← 0
        MOV     [SI+4],DX                   ;第2パラメータ ← DX=表示モード
        MOV     SI,G_MODE_SET               ;SI←モードセット
        CALL    GCALL                       ;ファンクション・コール
        POP     DS                          ;DS値を復元
        AND     AX,AX                       ;終了コードは正常か (AX=0) ?
        JZ      GMDRT                       ;正常であればGMDRTへ
        STC                                 ;異常終了(CY=1)とする
GMDRT:  RET                                 ;リターン
GMDST   ENDP                                ;プロシージャの終了

GTERM   PROC                                ;グラフィックスの終了プロシージャ定義
        CMP     BYTE PTR GSTSIN,1           ;グラフィックスシステムが開始されているか?
        JNE     GTERT                       ;開始されていなければGTERTへ
        MOV     GSTSIN,0                    ;グラフィックスシステムを終了とする
        PUSH    DS                          ;
        MOV     SI,G_TERM                   ;  } グラフィックスシステムを終了する
        CALL    GCALL                       ;
        POP     DS                          ;
GTERT:  RET                                 ;リターン
GTERM   ENDP                                ;プロシージャの終了
```

```
GDSET   PROC                                    ;DS:SI←パラメータ・ブロックの先頭アドレス
        MOV     AX,DTSEG
        MOV     DS,AX
        MOV     SI,OFFSET PBLOC
        RET
GDSET   ENDP

GCALL   PROC                                    ;SIで示されるファンクションをコールする
        PUSH    WORD PTR CS:GDSEG
        PUSH    WORD PTR CS:GDOFF
        LDS     BX,CS:GDADR
        SHL     SI,1
        SHL     SI,1
        CALL    DWORD PTR [BX+SI]
        RET
GCALL   ENDP

GSTSIN  DB      0
GDADR   DD      0
GDOFF   DW      0
GDSEG   DW      DTSEG
GOPER   DB      'グラフィックスドライバが未登録です。$'
MDSER   DB      '４０９６中１６色モード設定エラー$'

CODE    ENDS

DTSEG   SEGMENT PUBLIC                          ;パラメータ用セグメントの定義
PBLOC   DB      48 DUP(0)                       ;パラメータ・ブロック
WBLOC   DB      2000 DUP(0)                     ;ワーク・エリア
DTSEG   ENDS

        END
```

ここでの内容は、グラフィックス関係の各ファンクションのエントリーを登録したテーブルの先頭アドレスを取得し、これらのファンクションを呼び出すためのアドレスを知ることと、各ファンクションに対するパラメータを渡す時に使われるデータ領域を設定することです。なお、データ領域の構造はLIST 1-0で設定したように、システム側であらかじめ決められています。

LIST 1-0では、このデータ領域をセグメントDTSEGへ設定しています。また、GSTATでファンクションのエントリー・テーブルの先頭アドレスをラベル名GDADRへ取り込んでいます。そして、よく使われるファンクション専用に５つのプロシージャと、任意のファンクションを簡単に呼び出せるように２つのプロシージャ、計７つのプロシージャを用意しました。各プロシージャの内容は次の通りです。

GSTAT	グラフィック処理の初期設定及び開始
GTERM	グラフィック処理の終了
GDSET	DS:SIへパラメータ用アドレスを取得する
GCALL	SIへ設定したファンクションのコール
GMD16	640×400ドット、4096色中16色モードの設定
GMOD８	640×400ドット、8色中8色モードの設定
GM200	640×200ドット、8色中8色モードの設定

任意のファンクションを呼び出すために使われるプロシージャは、GDSETとGCALLですが、これらを利用したファンクション・コールの手順を示します。

1：	GDSETでDS:SIへパラメータ用アドレスを取得
2：	ファンクションが必要とする所定のパラメータをDS:[SI+0]番地から順番にセットする
3：	SIにファンクションNo.をセットしてGCALLをコールする

この手順に従って、ファンクションNo.14（画面クリア）をコールするGCLSを組んでみましょう(**LIST 1-1**)。

LIST 1-1

```
;*************************************
;*              LIST1-1              *
;*************************************

EXTRN   GSTAT:NEAR,GMD16:NEAR,GDSET:NEAR,GCALL:NEAR

CODE    SEGMENT PUBLIC

        ASSUME  CS:CODE,DS:CODE

PMAIN:  CALL    GSTAT                       ;グラフィックスの開始
        JB      TEXIT                       ;異常終了であればTEXITへ
        CALL    GMD16                       ;640×400,16色モード・セット
        JB      TEXIT                       ;異常終了であればTEXITへ
        CALL    GCLS                        ;画面クリア
TEXIT:  MOV     AX,4C00H                    ;リターン・コード・セット
        INT     21H                         ;MS-DOSへ

GCLS    PROC
        PUSH    DS                          ;DS値セーブ
        CALL    GDSET                       ;DS:SI←パラメータ・ブロックの先頭アドレス
        XOR     AX,AX                       ;AX←0
        MOV     [SI],AX                     ;第1パラメータ ← 0
        MOV     [SI+2],AX                   ;第2パラメータ ← 0
        MOV     SI,14                       ;画面消去用ファンクションNo.14を設定
        CALL    GCALL                       ;ファンクション・コール
        POP     DS                          ;DS値を復元
        RET                                 ;リターン
GCLS    ENDP

CODE    ENDS

STACK   SEGMENT STACK                       ;スタック・セグメントの定義
        DW      100H    DUP(0)
STACK   ENDS

        END
```

LIST 1-1では、LIST 1-0であらかじめ作っておいたプロシージャも使っています。それらは、EXTRNディレクティブでLIST 1-1の外部で定義したプロシージャであることを宣言しています。また、コードセグメント（命令コードが置かれているセグメント、本書ではCODEと命名）の定義では、セグメントのコンバインタイプにPUBLIC属性を指定していることに注意してください。MASMでは、PUBLIC属性が付けられた同じ名前を持ったセグメント（ここではCODE）を、1つの連続したセグメントとして結合してくれます。

なお、このコンバインタイプを省略した場合には、暗黙にPRIVATEが宣言されたことになります。したがって、省略した場合には、コードセグメント名が同じであっても、独立した別々のセグメントとして扱われます。

LIST 1-1では、画面モードをLIST 1-0で定義したプロシージャGMD16を使ってハイレゾリューション・モード（640×400ドット4096色中16色表示）に設定しましたが、8色中8色表示を選択する場合にはGMOD 8をコールしてください。

では、実際にLIST 1-0とLIST 1-1から実行ファイルを作ってみましょう。まず、LIST1-0とLIST1-1をそれぞれアセンブルして、オブジェクトファイルを作ります。そして、それぞれのオブジェクトファイルをリンクすることになります。このリンクの方法は、リンクしたいオブジェクトファイル名に先のオブジェクト・ファイル《LIST1-0.OBJ》を加えるということになります。

```
A>LINK   LIST1-1 LIST1-0;⏎
```

当然のことですが、本書のようにモジュール別のプログラム構造をとる場合には、MAKEファイルを作っておくと便利です。

```
《MAKE　ファイルの作り方》
＜目的ファイル＞:＜関連ファイル名＞････
            ＜コマンド＞
＜目的ファイル＞:＜関連ファイル名＞････
            ＜コマンド＞
                ⋮
                ⋮
```

MAKEファイルは分割アセンブルの必須のアイテムとなります。参考のために、LIST1-0.OBJとLIST1-1.OBJから実行ファイルLIST1-1.EXEを作るためのMAKEファイルのサンプルを示しておきます。

《 MAKEファイル LIST1-1の内容 》

```
LIST1-0.OBJ : LIST1-0.ASM
      MASM  LIST1-0;

LIST1-1.OBJ : LIST1-1.ASM
      MASM  LIST1-1;

LIST1-1.EXE : LIST1-1.OBJ LIST1-0.OBJ
        LINK  LIST1-1 LIST1-0;
```

MAKEファイルのファイル名は、慣例として開発プログラムと同じ名前を拡張子を付けずに用います。MAKEの実行に際しては、**MAKE.EXE**ファイルが必要です。先ほどのMAKEファイルの使用例は次のようになります。

```
A>MAKE  LIST1-1↵
```

さて、色々と準備をしてきましたが、これでグラフィック画面へのアクセスが可能となりました。次にグラフィック画面について、ビデオラムのアドレスマップとドットとの関係を見てみましょう。

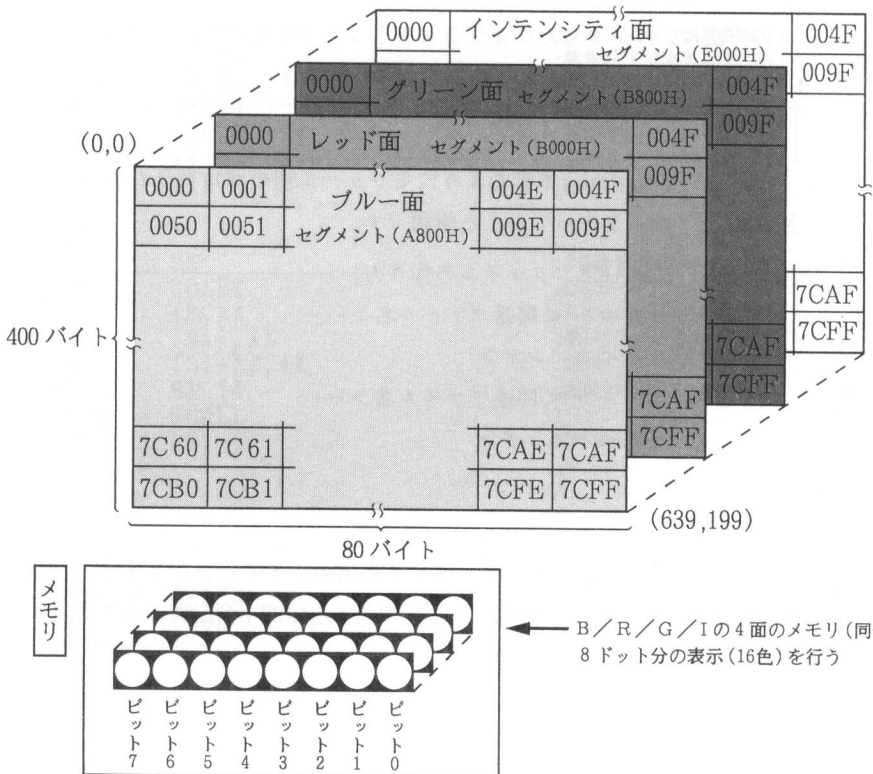



図 1-2-4 グラフィックVRAMのアドレスマップとドットの関係

図 1-2-4に示されているように、グラフィック画面はBRGI、4プレーンの同一オフセット・アドレスにおける1バイトで、8ドットを表します。したがって、8ドット単位でデータをセットする場合は単純ですが、ドット単位で操作する場合には他のドットに影響を与えないようプログラム側で考慮しなければなりません。例えば(0,0)の位置にカラー5で点を打つ場合、画面に何も表示されていなければB／G面の各オフセット・アドレス(0000H)に80H(10000000B)を単純に入れればいいのですが、何か絵が表示されているときはB／G面の各オフセット・アドレス(0000H)のビット7だけをセットし、R／I面のビット7をリセットするようにしなければならないわけです。

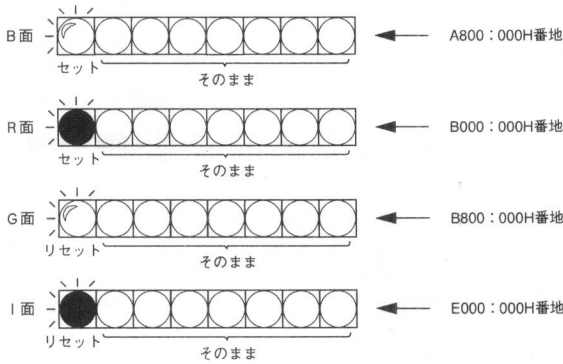



図 1-2-5 (0,0) にカラー5の点を打つ

これを実際にプログラム化したのがLIST 1-2です。選択した画面モードはハイレゾ・モード(4096色中16色表示)ですが、8色中8色を選択する場合にはI面に対するアクセスは不要となります。ここでは、各プレーンともオフセット・アドレス0000H番地のビット7をAND命令でリセットし、その後でカラーに合わせて、OR命令を実行しているということに注目してください。

LIST 1-2

```
;**************************************
;*               LIST1-2              *
;**************************************

EXTRN   GSTAT:NEAR,GMD16:NEAR

CODE    SEGMENT PUBLIC

        ASSUME  CS:CODE

DOTST   MACRO   VRSEG                   ;;ドットセット用マクロ定義
        LOCAL   DOTS1
        MOV     AX,VRSEG                ;;AX←VRSEG
        MOV     DS,AX                   ;;DS←VRSEG
        AND     [BX],DH                 ;;ビット・リセット
        ROR     CX,1                    ;;
        JNB     DOTS1                   ;;  } カラーに合わせドットをセットする
        OR      [BX],DL                 ;;
DOTS1:
        ENDM

PMAIN:  CALL    GSTAT                   ;グラフィックスの開始
        JB      TEXIT                   ;異常終了であればTEXITへ
```

```
        CALL    GMD16                   ;640×400,16色モード・セット
        JB      TEXIT                   ;異常終了であればTEXITへ
        MOV     CX,5                    ;カラーNO.セット
        MOV     DH,01111111B            ;
        MOV     DL,10000000B            ;
        MOV     BX,0                    ;
        DOTST   BLUE                    ;   指定カラーで点を打つ
        DOTST   RED                     ;
        DOTST   GREEN                   ;
        DOTST   ITSTY                   ;
TEXIT:  MOV     AX,4C00H                ;リターン・コード・セット
        INT     21H                     ;MS-DOSへ

CODE    ENDS

STACK   SEGMENT STACK                   ;スタック・セグメントの定義
        DW      100H    DUP(0)
STACK   ENDS

BLUE    SEGMENT AT 0A800H               ;B面のセグメントの定義
BLUE    ENDS

RED     SEGMENT AT 0B000H               ;R面のセグメントの定義
RED     ENDS

GREEN   SEGMENT AT 0B800H               ;G面のセグメントの定義
GREEN   ENDS

ITSTY   SEGMENT AT 0E000H               ;I面のセグメントの定義
ITSTY   ENDS

        END
```

ここでは単に点1つを打ったに過ぎませんが、実はこれは規模は小さくても「重ね合わせ」そのものです。最近のゲームでは重ね合わせ処理など当然のテクニックですが、その基本形がこの『ANDを取ってORを取る』ということなのです。つまり、ANDで存在しているものを一旦消去し、ORでそこに描き込むというわけです。

たかだかドット1つのプログラムですが、その延長線上にはあらゆる高度なテクニックが待ちかまえているのです。ちょうど、すべてのグラフィックスがドットの集まりであるかのように……。

本書では、添付のディスクに、ソースリストとアセンブル済みの実行ファイルが格納されていますので、プログラムはアセンブルすることなく実行／確認することができます。今回は、実行後自動的にMS-DOSシステムへ戻って来ますが、ループになっている場合は ESC キーによってMS-DOSシステムへ戻るようになっていますので、併せて覚えておいてください。

1－3　パレットいろいろ

手品とマシン語プログラム。一見すると何の脈絡もなさそうですが、実はちょっとした共通点があります。例えば、ごく普通のトランプを使った簡単な手品でも、タネがわからなければスゴイものに見えることがあります。同様に、マシン語プログラムにも相手の盲点をつくようなトリックがあるのです。もちろん、タネがわかれば「ナァ～ンダ!!」というようなことであるのは事実です。しかし、これが高度なテクニックと組み合わせられると、ついつい「もしかすると手品ではなく本物の魔法では……!?」などと思ってしまうから不思議なものです。

市販ソフトの中には、まるで PC-9801が特別な機能を隠していたかのごとくに、見事なテクニックを披露してくれるものもあります。そういった見事なものは、たいてい力まかせの本格プログラムですが、その裏には常に基本のトリックが潜んでいます。

ところで、パレットというと BASICでも簡単に実現できるため、どうしても軽視しがちです。なにしろ画面全体が一斉に変色するわけですから、下手に操作すれば誰にでも「アッ、これはパレットでチカチカさせているんだナ!!」とわかってしまうからです。

しかし、最初の節でパレットによるプレーン分割を紹介したように、パレットにはさまざまな利用法があります。しかも、一見すると「パレットなど何もいじっていない」というようなゲームソフトが、実はパレットをいじり回していると知ったら……。おちおちゲームで遊んでばかりはいられませんね。画面コントロールの基礎中の基礎テクニック、それがパレットの本当の姿です。

パレットをマシン語で操作するのは簡単です。カラー8色モードの場合、変更したいカラー番号を図 1-3-1に示されるポートへ出力すればいいのです。もちろん、グラフィックスドライバにもパレットを操作するファンクションがありますが、直接ポートを操作したほうが処理スピードも速く簡単です。

8色中8色モードを選択した場合、各ポートへの出力データは、上位4ビットと下位4ビットがそれぞれパレットとしての意味をもっています。つまり、1つのポートで2つのパレットを操作するようになっているのです。

ポートアドレス	パレット番号	
	上位4ビット	下位4ビット
A8H	3	7
AAH	1	5
ACH	2	6
AEH	0	4

図 1-3-1　8色中8色モード・カラーパレットのコントロール

では、これを実際に実行してみることにしましょう。プログラム（LIST 1-3）は、スペースキーを押すたびにパレット番号0を次々に変化させるものです。

LIST 1-3

```
;************************************
;*              LIST1-3             *
;************************************

EXTRN   GSTAT:NEAR,GTERM:NEAR,GMOD8:NEAR

CODE    SEGMENT PUBLIC

        ASSUME  CS:CODE,DS:CODE

PRINT   MACRO   STRING                      ;;文字列出力用マクロ定義
        MOV     DX,OFFSET STRING
        MOV     AH,09
        INT     21H
        ENDM

GETKEY  MACRO                               ;;1文字入力用マクロ定義
        LOCAL   GLOOP
GLOOP:  MOV     DL,0FFH
        MOV     AH,6
        INT     21H
        JZ      GLOOP
        ENDM

TXYSET  MACRO   REG,T_X,T_Y                 ;;座標設定用マクロ定義
        MOV     AX,TEXT
        MOV     DS,AX
        MOV     REG,&T_Y*160 + &T_X*2
        ENDM

HOMEC   EQU     1AH                         ; H.CLR キーのアスキーコード
ESCKY   EQU     1BH                         ; ESC キーのアスキーコード

PMAIN:  CALL    GSTAT                       ;グラフィックスの開始
        JB      TEXIT                       ;異常終了であればTEXITへ
        CALL    GMOD8                       ;640×400,8色モード・セット
        PRINT   TXCLR                       ;テキスト画面クリア
        PUSH    DS                          ;
        TXYSET  SI,38,12                    ;
        MOV     BYTE PTR [SI],30H           ;  カラー番号を画面中央に表示する
        MOV     BYTE PTR [SI+2000H],0E5H    ;
        POP     DS                          ;
TLOOP:  GETKEY                              ;
        CMP     AL,ESCKY                    ;  ESC キーが押されていればTEXITへ
        JE      TEXIT                       ;
        CMP     AL,HOMEC                    ;
        JNE     $+5                         ;  H.CLR が押されていれば
        CALL    DOKIC                       ;  DOKICをコール
        CMP     AL," "                      ;
        JNE     $+5                         ;  SPACE が押されていればPALOCをコール
        CALL    PALOC                       ;
        JMP     TLOOP                       ;TLOOPへ
TEXIT:  CALL    GTERM                       ;グラフィックス処理の終了とする
        PRINT   TKEEP                       ;テキスト画面の属性の復元
        MOV     AX,4C00H                    ;リターン・コード・セット
        INT     21H                         ;MS-DOSへ
```

```
PWSIGN  DB      0                              ;┐
PODATW  DB      0                              ;├ パレット用ワーク・エリア
PAL04   DB      0                              ;┘

PALOC   PROC
        MOV     AL,PWSIGN                      ;┐
        OR      AL,AL                          ;├ ALの値によって垂直同期を取る
        JE      $+5                            ;│
        CALL    VWAIT                          ;┘
        MOV     AL,PAL04                       ;┐
        ADD     AL,10H                         ;│
        CMP     AL,84H                         ;│
        JNE     PALST                          ;├ パレット0を変化させる
        MOV     AL,04H                         ;│
PALST:  MOV     PAL04,AL                       ;│
        OUT     0AEH,AL                        ;│
        RET                                    ;┘
PALOC   ENDP

DOKIC   PROC
        XOR     PWSIGN,1                       ;┐
        PUSH    AX                             ;│
        PUSH    DS                             ;│
        TXYSET  SI,38,12                       ;├ 垂直同期を取るか否かのフラグを変更する
        XOR     BYTE PTR [SI],1                ;│
        POP     DS                             ;│
        POP     AX                             ;│
        RET                                    ;┘
DOKIC   ENDP

VWAIT   PROC
        IN      AL,0A0H                        ;┐
        TEST    AL,00100000B                   ;│
        JNE     VWAIT                          ;│
VWAT2:  IN      AL,0A0H                        ;├ 垂直同期期間を取る
        TEST    AL,00100000B                   ;│
        JE      VWAT2                          ;│
        RET                                    ;┘
VWAIT   ENDP

TXCLR   LABEL BYTE                             ;テキスト・クリア & 文字の属性の保存
        DB      1BH,"[2J"
        DB      1BH,"[s",1BH,"[>5h",1BH,"[1>h$"

TKEEP   LABEL BYTE                             ;文字の属性の復元
        DB      1BH,"[u",1BH,"[>5l",1BH,"[1>l$"

CODE    ENDS

STACK   SEGMENT STACK                          ;スタック・セグメントの定義
        DW      100H    DUP(0)
STACK   ENDS

TEXT    SEGMENT AT 0A000H                      ;テキスト用セグメントの定義
TEXT    ENDS

        END
```

ここでは、８色モードを使いますから、プログラムの先頭で画面モードを８色中８色に設定するGMOD８をコールしています。

```
A＞LIST1-3 ⏎
```

実行してみると、画面中央付近に '0' と表示され、スペースキーと連動してパレット0が変わります。ここでスペースキーを速射砲のように連打してみてください。パレットの変化に伴い、画面を寸断するようなチラツキが気になってきます。これは、パソコンのディスプレイが**図 1-3-2**のような順で1/60秒毎に表示を繰り返しているため、表示の途中でパレットを変更すると瞬間的に画面の上下で色が変わってしまうからです。

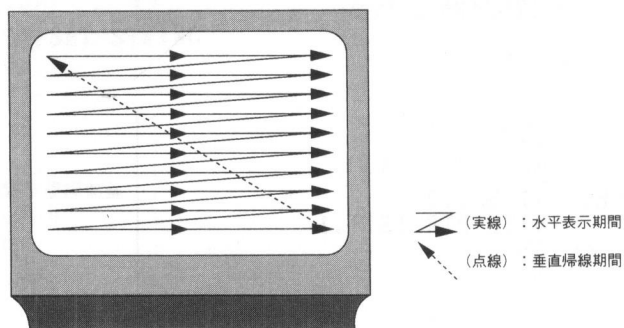



図 1-3-2　ハードウェアから見た画面表示の実体

これを避けるためには、表示が完了して次の表示が始まるまでの間（走査線が上にもどる垂直帰線期間内）にパレット変更をすればいいのです。

今度は H.CLR キーを押して同じようにしてみてください。画面中央には違いを示すため '1' と表示され、パレット変化の際のチラツキはなくなります。本来ならば、グラフィックVRAMをアクセスするときにもこのように同期を取るべきなのですが、特に問題となるケース以外は行いません。

なお、本プログラムでは２度目の垂直帰線期間をポーリング（サーチ）することで、正確な垂直帰線のスタートを得ていましたが、実は1/60秒毎のV-SYNC割り込みとはこの垂直帰線のスタートで発生する割り込みのことなのです。したがって、V-SYNC割り込みでパレットを変化させる場合は、このような気を使う必要はありません。

次に、アナログ 4096色モードでのパレット変更を行ってみましょう。アナログに使用するポートは**図 1-3-1**と同じですが、操作方法が異なります。まず、ポートA8HへパレットNo.（0～15）を出力します。次に、そのパレットに対する緑の輝度をポートAAHへ、赤の輝度をポートACHへ、青の輝度をポートAEHへそれぞれ出力するようになっています。なお、各輝度は16階調で設定します。

I/Oポート アドレス	制御データ								内　容
	b7	b6	b5	b4	b3	b2	b1	b0	
A 8H	0	0	0	0	I	G	R	B	選択したパレットNo.
AAH	0	0	0	0	G3	G2	G1	G0	緑の輝度を16階調で設定
ACH	0	0	0	0	R3	R2	R1	R0	赤の輝度を16階調で設定
AEH	0	0	0	0	B3	B2	B1	B0	青の輝度を16階調で設定

図 1-3-3　アナログ・カラーのポート操作

プログラム(**LIST 1-4**)は、先ほどと同じような感覚でパレット0を輝度別に変化させるものです。それぞれの輝度は、青がテンキーの[7]－[9]、赤がテンキーの[4]－[6]、そして、緑がテンキーの[1]－[3]で連続して調整することができます。また、垂直帰線の同期は [H.CLR] でオン／オフができます。

LIST 1-4

```
;*************************************
;*            LIST1-4               *
;*************************************

EXTRN   GSTAT:NEAR,GTERM:NEAR,GMD16:NEAR

CODE    SEGMENT PUBLIC

        ASSUME  CS:CODE,DS:CODE

PRINT   MACRO   STRING                          ;;文字列出力用マクロ定義
        MOV     DX,OFFSET STRING
        MOV     AH,09
        INT     21H
        ENDM

GETKEY  MACRO                                   ;;1文字入力マクロ定義
        LOCAL   GLOOP
GLOOP:  MOV     DL,0FFH
        MOV     AH,6
        INT     21H
        JZ      GLOOP
        ENDM

TXYSET  MACRO   REG,T_X,T_Y                     ;;テキスト文字ダイレクト表示マクロ定義
        MOV     AX,TEXT
        MOV     DS,AX
        MOV     REG,&T_Y*160 + &T_X*2
        ENDM

                                                ; [H.CLR] キーのアスキーコード
HOMEC   EQU     1AH
ESCKY   EQU     1BH                             ; [ESC] キーのアスキーコード

PMAIN5: CALL    GSTAT                           ;グラフィックスの開始
        JB      TEXIT                           ;異常終了であればTEXITへ
        CALL    GMD16                           ;640×400,16色モード・セット
        JB      TEXIT                           ;異常終了であればTEXITへ
        PRINT   TXCLR                           ;テキスト画面クリア
        PUSH    DS
        TXYSET  SI,0,0                          ;1
```

```
        MOV     BYTE PTR [SI],"V"           ;
        MOV     BYTE PTR [SI+2],"="         ;
        MOV     BYTE PTR [SI+4],"0"         ;
        MOV     BYTE PTR [SI+160],"B"       ;
        MOV     BYTE PTR [SI+162],"="       ; テキスト画面初期設定
        MOV     BYTE PTR [SI+320],"R"       ;
        MOV     BYTE PTR [SI+322],"="       ;
        MOV     BYTE PTR [SI+480],"G"       ;
        MOV     BYTE PTR [SI+482],"="       ;
        POP     DS                          ;
        MOV     AX,CS                       ;
        MOV     ES,AX                       ;
        CALL    DTDIP                       ;
        CLD                                 ;
TLOOP:  GETKEY                              ;ディレクション・フラグ・クリア
        CMP     AL,ESCKY                    ;1文字入力
        JE      TEXIT                       ; [ESC] キーが押されたか?
        CMP     AL,HOMEC                    ;押されていればTEXITへ
        JNE     $+5                         ;
        CALL    DOKIC                       ; [H.CLR] キーが押されてればDOKIC
        CALL    ACHAN                       ; をコール
        JNE     $+5                         ;B,R,Gの輝度値をチェックする
        CALL    PALOC                       ;輝度変化ありか?
        JMP     TLOOP                       ;ありならパレットを変更する
TEXIT:  CALL    GTERM                       ;TLOOPへ
        PRINT   TKEEP                       ;グラフィック処理の終了
        MOV     AX,4C00H                    ;テキスト画面の属性の復元
        INT     21H                         ;リターン・コード・セット
                                            ;MS-DOSへ
ANABB   DB      0
ANARR   DB      0                           ;青色の輝度値
ANAGG   DB      0                           ;赤色の輝度値
                                            ;緑色の輝度値
KEYTBL  LABEL BYTE
        DB      "7","4","1","9","6","3"     ;キー情報テーブル
ADRTBL  LABEL BYTE                          ;キー情報保存エリア選択用
        DW      ANAGG,ANARR,ANABB,ANAGG,ANARR,ANABB

ACHAN   PROC
        MOV     CX,6                        ;
        MOV     DI,OFFSET KEYTBL            ;
        REPNZ   SCASB                       ;
        JNE     ACHRT                       ;
        MOV     BX,OFFSET ADRTBL            ;
        ADD     BX,CX                       ;
        ADD     BX,CX                       ;
        MOV     BX,[BX]                     ;
        MOV     AL,[BX]                     ;
        CMP     CX,3                        ; [4]~[6]が押されていれば輝度値
        JGE     ACHA1                       ;
        INC     AL                          ; を変更する
        CMP     AL,16                       ;
        JL      ACHA2                       ;
        INC     AX                          ;
        JMP     ACHRT                       ;
ACHA1:  DEC     AL                          ;
        JS      ACHRT                       ;
ACHA2:  MOV     [BX],AL                     ;
        XOR     AX,AX                       ;
ACHRT:  RET                                 ;
ACHAN   ENDP                                ;
```

```
PWSIGN  DB      0                                  ;
PODATW  DB      0                                  ;} パレット用ワーク・エリア
PAL04   DB      0                                  ;

TDTSET  MACRO   ADR,ADD                            ;;アスキーコード変換用マクロ定義
        LOCAL   DTDI1,DTDI2
        MOV     AL,CS:ADR
        CMP     AL,10
        JGE     DTDI1
        OR      AL,30H
        JMP     DTDI2
DTDI1:  SUB     AL,9
        OR      AL,40H
DTDI2:  MOV     [SI+ADD],AL
        ENDM

PALOC   PROC
        MOV     AL,PWSIGN                          ;
        OR      AL,AL                              ;} ALの値によって垂直同期を取る
        JE      $+5                                ;
        CALL    VWAIT                              ;
        XOR     AL,AL                              ;
        OUT     0A8H,AL                            ;
        MOV     AL,ANAGG                           ;
        OUT     0AAH,AL                            ;
        MOV     AL,ANARR                           ;} アナログ・パレットの変更
        OUT     0ACH,AL                            ;
        MOV     AL,ANABB                           ;
        OUT     0AEH,AL                            ;
DTDIP:  PUSH    DS                                 ;
        TXYSET  SI,2,1                             ;
        TDTSET  ANABB,0                            ;
        TDTSET  ANARR,160                          ;} 各面の輝度値を表示する
        TDTSET  ANAGG,320                          ;
        POP     DS                                 ;
        RET                                        ;
PALOC   ENDP

DOKIC   PROC
        XOR     PWSIGN,1                           ;
        PUSH    AX                                 ;
        PUSH    DS                                 ;
        TXYSET  SI,2,0                             ;} 垂直同期を取るか否かのフラグを変更する
        XOR     BYTE PTR [SI],1                    ;
        POP     DS                                 ;
        POP     AX                                 ;
        RET                                        ;
DOKIC   ENDP

VWAIT   PROC
        IN      AL,0A0H                            ;
        TEST    AL,00100000B                       ;
        JNE     VWAIT                              ;
VWAT2:  IN      AL,0A0H                            ;} 垂直同期期間を得る
        TEST    AL,00100000B                       ;
        JE      VWAT2                              ;
        RET                                        ;
VWAIT   ENDP
                                                   ;テキスト・クリア & 文字の属性の保存
TXCLR   LABEL BYTE
        DB      1BH,"[2J"
        DB      1BH,"[s",1BH,"[>5h",1BH,"[1>h$"
```

```
TKEEP   LABEL BYTE                              ;文字の属性の復元
        DB      1BH,"[u",1BH,"[>51",1BH,"[1>1$"

CODE    ENDS

STACK   SEGMENT STACK                           ;スタック・セグメントの定義
        DW      100H    DUP(0)
STACK   ENDS

TEXT    SEGMENT AT 0A000H                       ;テキスト用セグメントの定義
TEXT    ENDS

        END
```

画面上部には、垂直同期オン／オフの状態（1でオン）とパレット0の緑、赤、青のそれぞれの輝度が表示されます。こうして連続して輝度を変化させると、垂直同期を取る重要性が改めて感じられるかもしれません。

V=0	…垂直同期の状態
B=0	…青の輝度レベル
R=0	…赤の輝度レベル
G=0	…緑の輝度レベル

アナログカラーは 新しく追加された機能ですから、旧タイプのようにプレーンが3枚の機種では当然対応できません。無理に実行すれば、ポートへ出力するデータ構成が全く異なりますから、期待通りの画面とはかけ離れたものとなります。このような場合は、「アナログ専用」としてしまうのが普通です。このプログラム（**LIST 1-4**）では、何もせずに MS-DOSへ戻るようにしています。

では、PC-9801本体がアナログ対応であってもディスプレイがアナログに対応していない場合はどうでしょうか。プログラムでいちいち区別するのでは面倒です。しかし、実はこれは心配することはありません。ディスプレイがデジタルの場合は、輝度レベルが半分より小（0～7）であればその色の輝度は0となり、半分より大であれば、輝度15（最大）で発色するようになっているのです。すなわち、自動的に中間輝度が省かれるのです。どうしても特殊な中間色で固定したい場合は、「要アナログ・ディスプレイ」としなければなりませんが、実際のゲームでは8色でもおかしくない色に設定するか、中間色を単にパレットのなめらかな変化に利用して、デジタル・ディスプレイでも支障のないようにしておくのが一般的です。こうすれば、アナログ・ディスプレイの方はより以上に美しい画面で、デジタル・ディスプレイの方はそれなりの画面でゲームを楽しむことができるからです。

パレットの具体的な応用例については別の章で示しますが、プレーンの状態とカラー番号との関係（**図 1-1-3**参照）を熟知しておくことが最大のキーポイントです。パレットをいろいろと変化させながら、カラー番号（0～15）を聞いただけでプレーンの状態（オンかオフか）がすぐわかるよう慣れ親しんでください。

第2章　EGC拡張グラフィックスのすべて

『斬り捨て御免……』

　こんな言葉は江戸時代のことかと思っていたら、なんとこの近代社会においても『足切り』などという妙なモノに姿を変えて残っているようです。時代は違えど、切られる側はたまったものではありません。とはいえ、現代は自由競争の社会。勝者がいれば必ず敗者がでる仕組みなのであります。

　無差別に『差別』をするのは問題ですが、ヤヤコシイことに『差別』と『区別』は区別をしなければなりません。例えば、ホテルや旅館の大浴場。男風呂と女風呂とを分けるのは区別でも、男風呂が大きくて女風呂が小さいと「これは差別だ!!」と問題になるわけです。ところで……。

『EGC 対応機種専用』というのは差別か区別か……。私には区別ができませんが、少なくともそれまでの機種との間に違いがあるのは事実です。その事実を正確に認識することは、PC-9801VXより前のユーザーにとってもけっしてマイナスではないと思うのですが……。この章は、『EGC対応機種専用』です。

2－1　ＧＲＣＧ互換モード

よく言葉が一人歩きするといいますが、EGC（Enhanced Graphic Charger）もPC-9801VX以降になって急激に脚光を浴びた名称です。EGCは、PC-9801U／UV／VF／VM（PC-9801Uではオプション）に搭載されたGRCG（GRaphic CharGer）の機能を拡張したもので、PC-9801VX以降搭載された機能です。最近のゲームソフトでは、このEGC対応版もかなり増えてきましたが、EGCには次にあげる4つの主な機能があります。

① GRCGとの互換モードを有する。
② 4プレーンの同時アクセスが可能。
③ CPUのデータとVRAMデータとのビット・パターン間の演算（ラスターオペレーション：ROP）機能を有する。
④ データ移動におけるビット単位のシフトが可能。

この節では、GRCGとEGCのGRCG互換モードに焦点をあてていきます。グラフィックスシステムの初期状態では、GRCGやEGC搭載機種であっても通常のアクセス・モードがセレクトされていますから、まずはこのシステムのモードを切り換えてGRCGを使用可能な状態にしなければなりません。GRCGには、TDWモード、TCRモード、RMWモードの3つのモードがあり、ポート7CH, 7EHを介してコントロールします（図 2-1-1）。

命令	ポートアドレス	b7	b6	b5	b4	b3	b2	b1	b0
ライトモードレジスタ	7CH	CG モード	RMW モード	0	0	PIEN	PGEN	PREN	PBEN
ライトタイルレジスタ	7EH	タイルレジスタ PB〜PI							

(注) モードレジスタにライトを行うと、タイルレジスタはB面用のタイルレジスタがセレクトされ、その後ライトする毎に、レジスタR,G,Iと順に変化する。また、ワードアクセス時は、16ビットに拡張される。なお、モードライト時は割り込みを禁止しなければならない。

ビット名	ビット＝1の意味	ビット＝0の意味
CGモード	・GRCGを有効にする ・CPUのVRAMアクセスをきっかけとしてGRCGが各モードの動作を実行する	・GRCGを無効とする ・CPUのVRAMアクセスは、そのままVRAMのリード（ライト）となる
RMWモード	・CPUのVRAMライトによりRMWモードの動作を行う ・CPUのVRAMリードは無視される	・CPUのVRAMライトによりTDWモードの動作を行う ・CPUのVRAMリードによりTCRモードの動作を行う
PB PR PG PI	・該当するプレーンを無効とする	・該当するプレーンを有効とする ・複数ビットの指定が可能 ・GRCGは有効となっているプレーンに対してのみアクセスを行う

・TDWモード
CPUがVRAMへ書き込み動作をすると、あらかじめ設定してあったタイルレジスタの内容を各プレーンに対して書き込む。CPUのデータは無視される。

・TCRモード
CPUがVRAMの該当アドレスをリードすると、各プレーンとタイルレジスタの一致をとり、すべてのプレーンで一致のとれたビットを1にしてCPUにリードデータとして出力する。

・RMWモード
CPUがVRAMにデータを書き込むと、CPUのライトデータの内、1となっているビットに対するタイルレジスタPB〜PIの内容が各プレーンにライトされる。0のビットに対するVRAMデータは変化しない。

図 2-1-1 グラフィックチャージャ（GRCG）のI／Oコントロール

これらが、GRCGをコントロールするためのポートです。従来のGRCGでは、CPUのVRAMアクセスに限るという条件がありましたが、EGCの互換機能では、GDC（Graphic Display Controller：LSI μPD7220A）のVRAMアクセスに対しても有効になっています（GDCに関しては後述します）。

ここでは、参考としてTDWモードを使って指定したタイルパターンで全画面を埋めるプログラムを示します（LIST 2-0）。なお、タイルパターンは次のようなキー操作で変更が可能となっています。

・画面のクリア　　　　　　　　　・・・・　[H.CLR] キー
・プレーンBのタイルパターンの変更　・・・・　テンキーの[7],[9]
・プレーンRのタイルパターンの変更　・・・・　テンキーの[4],[6]
・プレーンGのタイルパターンの変更　・・・・　テンキーの[1],[3]
・プレーンＩのタイルパターンの変更　・・・・　テンキーの[0],[.]

LIST 2-0

```
;************************************
;*              LIST2-0             *
;************************************

EXTRN   GSTAT:NEAR,GTERM:NEAR,GMD16:NEAR

CODE    SEGMENT PUBLIC

        ASSUME  CS:CODE,DS:CODE,SS:STACK

PRINT   MACRO   STRING                      ;;文字列出力用マクロ定義
        MOV     DX,OFFSET STRING
        MOV     AH,09
        INT     21H
        ENDM

GETKEY  MACRO                               ;;1文字入力マクロ定義
        LOCAL   GLOOP
GLOOP:  MOV     DL,0FFH
        MOV     AH,6
        INT     21H
        JZ      GLOOP
        ENDM

HOMEC   EQU     1AH                         ; [H.CLR] キーのアスキーコード
ESCKY   EQU     1BH                         ; [ESC] キーのアスキーコード

TXYSET  MACRO   REG,T_X,T_Y                 ;;テキスト画面ダイレクト表示マクロ定義
        MOV     AX,TEXT
        MOV     DS,AX
        MOV     REG,&T_Y*160 + &T_X*2
        ENDM

OUTAL   MACRO   PORT,DATA                   ;;PORTへDATAをOUTする
        MOV     AL,DATA
        OUT     PORT,AL
        ENDM

BEGIN:  CALL    GSTAT                       ;グラフィックスの開始
        JNB     NOTTX                       ;異常終了であればTEXITへ
        JMP     TEXIT                       ;
```

```
NOTTX:  CALL    GMD16                       ;640×400,16色モード・セット
        JB      TEXIT                       ;異常終了であればTEXITへ
        PRINT   TXCLR                       ;テキスト画面クリア
        PUSH    DS                          ;┐
        TXYSET  SI,0,0                      ;│
        MOV     BYTE PTR [SI+160],"B"       ;│
        MOV     BYTE PTR [SI+162],"="       ;│
        MOV     BYTE PTR [SI+164],"0"       ;│
        MOV     BYTE PTR [SI+166],"0"       ;│
        MOV     BYTE PTR [SI+320],"R"       ;│
        MOV     BYTE PTR [SI+322],"="       ;│
        MOV     BYTE PTR [SI+324],"0"       ;├ テキスト画面の初期設定
        MOV     BYTE PTR [SI+326],"0"       ;│
        MOV     BYTE PTR [SI+480],"G"       ;│
        MOV     BYTE PTR [SI+482],"="       ;│
        MOV     BYTE PTR [SI+484],"0"       ;│
        MOV     BYTE PTR [SI+486],"0"       ;│
        MOV     BYTE PTR [SI+640],"I"       ;│
        MOV     BYTE PTR [SI+642],"="       ;│
        MOV     BYTE PTR [SI+644],"0"       ;│
        MOV     BYTE PTR [SI+646],"0"       ;│
        POP     DS                          ;┘
        CLD                                 ;ディレクション・フラグ・クリア
TLOOP:  GETKEY                              ;┐
        CMP     AL,ESCKY                    ;├ [ESC] キーが押されたらTEXITへ
        JE      TEXIT                       ;┘
        CMP     AL,HOMEC                    ;┐
        JNE     TEST1                       ;├ [H.CLR] キーが押されたら画面をクリアする
        CALL    GVCLR                       ;│
        JMP     TLOOP                       ;┘
TEST1:  CALL    TCHAN                       ;┐ タイル・パターンの変更
        JMP     TLOOP                       ;┘
TEXIT:  CALL    GTERM                       ;グラフィックス処理の終了とする
        PRINT   TKEEP                       ;テキスト画面の属性の復元
        MOV     AX,4C00H                    ;リターン・コード・セット
        INT     21H                         ;MS-DOSへ

TILP0   DB      0                           ;┐
TILP1   DB      0                           ;├ タイル・パターンの保存
TILP2   DB      0                           ;│
TILP3   DB      0                           ;┘

TCHAN   PROC
        MOV     CX,CS                       ;┐
        MOV     ES,CX                       ;│
        MOV     CX,8                        ;│
        MOV     DI,OFFSET KEYTBL            ;│
        REPNZ   SCASB                       ;│
        JNE     TCHRT                       ;│
        MOV     BX,OFFSET ADRTBL            ;│
        ADD     BX,CX                       ;├ キー入力に従ってタイル・パターンを変更
        ADD     BX,CX                       ;│ する
        MOV     BX,[BX]                     ;│
        CMP     CX,4                        ;│
        JGE     TCHA1                       ;│
        INC     BYTE PTR [BX]               ;│
        JMP     TCHA2                       ;│
TCHA1:  DEC     BYTE PTR [BX]               ;│
TCHA2:  CALL    TBXFL                       ;│
TCHRT:  RET                                 ;┘
TCHAN   ENDP
```

```
KEYTBL  LABEL BYTE                          ;キー情報テーブル
        DB      "7","4","1","0"
        DB      "9","6","3","."

ADRTBL  LABEL WORD                          ;タイル・パターン保存エリア選択用テーブル
        DW      TILP3,TILP2,TILP1,TILP0
        DW      TILP3,TILP2,TILP1,TILP0

TDISP   PROC
        MOV     AL,CS:[DI]                  ;
        MOV     CL,4                        ;
        SHR     AL,CL                       ;
        CALL    TDIS0                       ;
        ADD     SI,2                        ;
        MOV     AL,CS:[DI]                  ; タイル・パターンの表示
        AND     AL,0FH                      ;
        CALL    TDIS0                       ;
        ADD     SI,158                      ;
        RET                                 ;
TDISP   ENDP

TDIS0   PROC
        CMP     AL,10                       ;
        JGE     TDIS1                       ;
        OR      AL,30H                      ;
        JMP     TDIS2                       ;
TDIS1:  SUB     AL,9                        ; ALの数値をアスキーコードへ変換する
        OR      AL,40H                      ;
TDIS2:  MOV     [SI],AL                     ;
        RET                                 ;
TDIS0   ENDP

TBXFL   PROC
        MOV     BX,WORD PTR TILP0           ;
        MOV     CX,WORD PTR TILP2           ;
        CALL    BOXFL                       ;
        PUSH    DS                          ;
        TXYSET  SI,2,1                      ;
        MOV     DI,OFFSET TILP0             ;
        CALL    TDISP                       ;
        MOV     DI,OFFSET TILP1             ; それぞれのタイル・パターンで画面表示
        CALL    TDISP                       ;
        MOV     DI,OFFSET TILP2             ;
        CALL    TDISP                       ;
        MOV     DI,OFFSET TILP3             ;
        CALL    TDISP                       ;
        POP     DS                          ;
        RET                                 ;
TBXFL   ENDP

GVCLR   PROC
        XOR     BX,BX                       ;
        MOV     CX,BX                       ; 画面クリア
        CALL    BOXFL                       ;
        RET                                 ;
GVCLR   ENDP

BOXFL   PROC
        CLI                                 ;
        OUTAL   7CH,080H                    ;
        STI                                 ;
        OUTAL   7EH,BL                      ;
        OUTAL   7EH,BH                      ;
        OUTAL   7EH,CL                      ;
```

```
        OUTAL   7EH,CH                          ;
        MOV     CX,7D00H/2                      ;  指定タイルパターンで画面表示
        MOV     AX,BLUE                         ;
        MOV     ES,AX                           ;
        XOR     DI,DI                           ;
        REP     STOSW                           ;
        CLI                                     ;
        OUTAL   7CH,0                           ;
        STI                                     ;
        RET                                     ;
BOXFL   ENDP

TXCLR   LABEL BYTE                              ;テキスト・クリア  &  文字の属性の保存
        DB      1BH,"〔2J"
        DB      1BH,"〔s",1BH,"〔〉5h",1BH,"〔1〉h$"

TKEEP   LABEL BYTE                              ;文字の属性の復元
        DB      1BH,"〔u",1BH,"〔〉5l",1BH,"〔1〉l$"

CODE    ENDS

STACK   SEGMENT STACK                           ;スタック・セグメントの定義
        DW      100H    DUP(0)
STACK   ENDS

TEXT    SEGMENT AT 0A000H                       ;テキスト用セグメントの定義
TEXT    ENDS

BLUE    SEGMENT AT 0A800H                       ;B面用セグメントの定義
BLUE    ENDS

        END
```

GRCGは４画面同時アクセスが可能ですから、かなり高速な処理をしてくれます。しかし、TDWモードにしても、RMWモードにしても、VRAMへ書き込まれるデータはあらかじめポート7EHへ設定しておいたタイルパターンが関与することに変わりありません。そのため、タイルパターンとVRAMデータとの間でビットパターン間の演算（ラスターオペレーション：以後ROPと記述）をしたい場合は、ROP後のデータをいちいちポート7EHへ設定しなければなりません。これでは、４画面同時アクセスの意味がなくなってしまいます。

このような問題点を一挙に解決してくれるのが、次のテーマであるEGCの拡張モードというわけです。

2－2　EGCの拡張モード

かなりの期待を抱かせながら新しい節へと移ってきたわけですが、このEGCの拡張モードもGRCGの互換モードと同じように、拡張モードへと切り換えることで使用可能になります。この切り換えは、ポート 6AHのビット0,2を1に、7CHのCGビットを1にして行います。

ポートアドレス	モード	b7	b6	b5	b4	b3	b2	b1	b0
6AH	互換モード	0	0	0	0	0	1	0	0
	拡張モード	0	0	0	0	0	1	0	1

ポートアドレス	モード	b7	b6	b5	b4	b3	b2	b1	b0
7CH	ノーマルモード	0	0	0	0	0	0	0	0
	互換モード	1	0	0	0	0	0	0	0
	拡張モード	1	0	0	0	0	0	0	0

(注) 拡張モードから他のモードへ移行する場合には、必ずポート4A0Hへ FFF0Hをセットしなければなりません。また、モードチェンジは割り込み禁止の状態で行います。

図 2-2-1 EGC描写モードのI／Oコントロール

この節では全機能を一覧できるようにしていますが、ここですべてを覚えることはありません。必要に応じて、そのつど確認すればいいのです。本書でも、次節からは具体的な利用法が出てきます。

さて、拡張モードでグラフィックVRAMを操作するための I／Oポートですが、これには 4A0H～4AEHの8つがあります。図 2-2-2を参照してください。

図 2-2-2 EGC (Enhanced Graphic Charger) のI／Oコントロール

ポート＼アドレス	b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
4A0H	1	1	1	1	1	1	1	1	1	1	1	1	アクティブプレーン PI	PG	PR	PB
4A2H	0	FG	BG	0	リードプレーン				1	1	1	1	1	1	1	1
4A4H	モードレジスタ								ROPコード							
4A6H	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4A8H	マスクレジスタ：全プレーンに対し１つのみ対応。 ROPに優先される															
4AAH	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4ACH	0	0	0	*	0	0	0	0	ディスティネーションビットアドレス				ソースビットアドレス			
4AEH	0	0	0	0	ビット長											

＊：転送方向の設定。0で＋方向、1で－方向となる

FGまたは、BG＝1の時

ポート	b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
4A6H	0	0	0	0	0	0	0	0	0	0	0	0	フォアグラウンドカラー			
4A8H	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4AAH	0	0	0	0	0	0	0	0	0	0	0	0	バックグラウンドカラー			

※アクティブ・プレーン

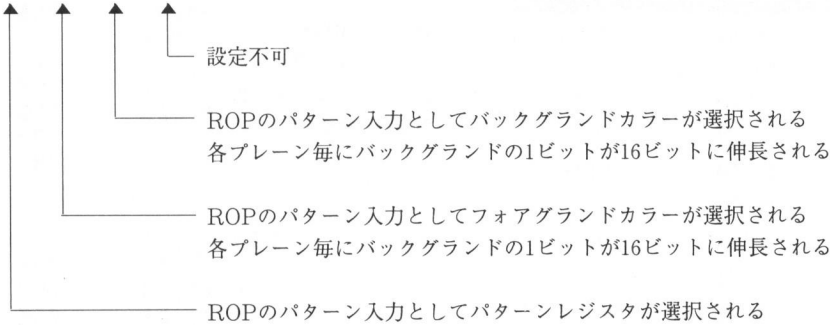
命令が適用されるプレーンをビット対応で設定する。すべての命令に優先する。

ライト系：”1”のプレーンは変更されない。

リード系：コンペアリードビット＝１の時に有効。”1”のプレーンはコンペアされない。

※リードプレーン

コンペアビット＝0の時に読み込むプレーンを設定する。

※FGC,BGC

※マスクレジスタ

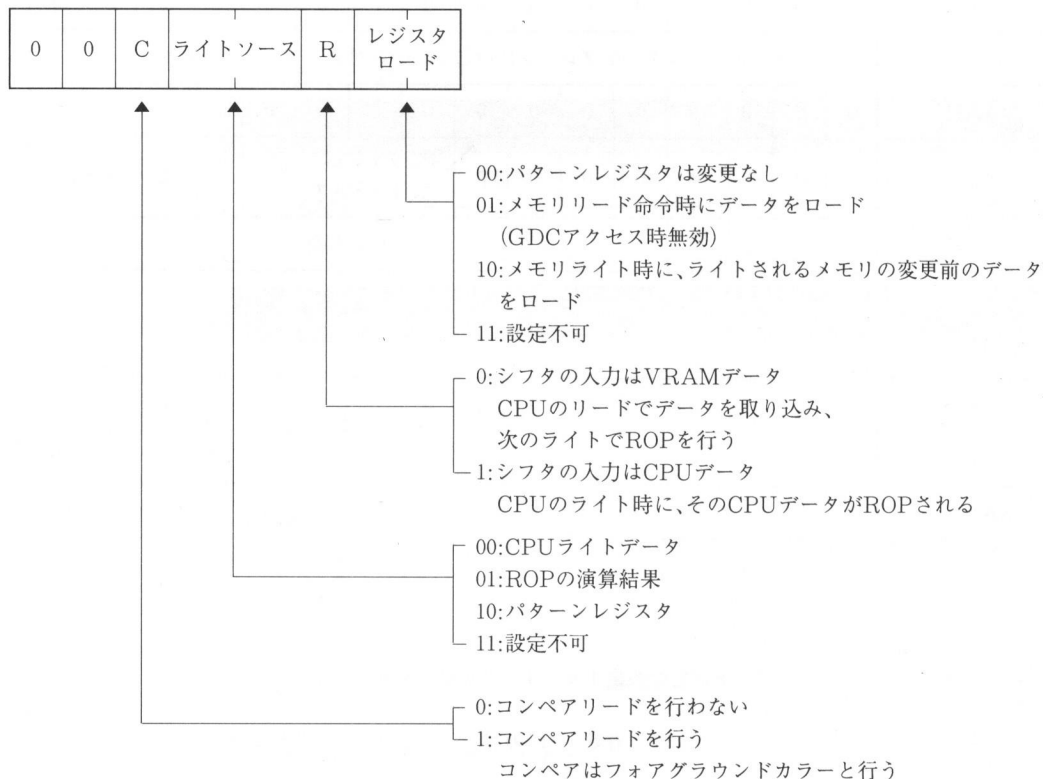
全プレーンに対してマスクレジスタは１つだけ存在する。
0のビットはデータ書き込み後も変更されない。
ROPに優先される。

※フォアグラウンドカラー

FGC指定時のフォアグラウンドカラーを設定する。

※バックグラウンドカラー

BGC指定時のバックグラウンドカラーを設定する。

※モードレジスタ

※ＲＯＰコードレジスタ

ROPの演算内容をビットパターンにより設定する。
S:入力、D:(VRAM上のデータ)とすると16種類のROP演算が可能。
代表的なROPコードは次のとおり。

演算例	ROPコードA	ROPコードB
S	0F0H	0AAH
$\overline{S}$	0FH	
D	0CCH	0CCH
$\overline{D}$	033H	33H
$D+S$	0FCH	0EEH
$D+\overline{S}$	0CFH	
$\overline{D}+S$	0F3H	0BBH
$\overline{D}+\overline{S}$	03FH	
$D \cdot S$	0C0H	88H

＋　：OR　(論理和)
・　：AND (論理積)
(＋)：XOR (排他的論理和)
－　：NOT (反転)

D・$\overline{S}$	0CH	
$\overline{D}$・S	30H	22H
$\overline{D}$・$\overline{S}$	03H	
D (+) S	03CH	66H
D (+) $\overline{S}$	0C3H	
$\overline{D}$ (+) S	0C3H	99H
$\overline{D}$ (+) $\overline{S}$	03CH	

(注) ROPコードBに対する $\overline{S}$ は、Sと同じコードを選択し、カラーの設定で対応する。

※ソースビットアドレス、ディスティネーションアドレス

データを転送する場合のワード内のビットアドレスを設定する(GDCアクセス時は無効)。

＊ ＝ 0:ワードの先頭から数える

＊ ＝ 1:ワードの後方から数える

※ビット長

転送するビット長を設定する。1ラスタ方向のみ。内容は保存される。

ポート4A0H～4AEHへ所定のデータを送ると、EGCの拡張モードが使えるようになります。以後VRAMへアクセスすることによって、指定した動作が可能となるわけです。ではここで、EGCの拡張モードをオン／オフするプロシージャを定義してみましょう(LIST 2-1)。

LIST 2-1

```
;************************************
;*              LIST2-1             *
;************************************

PUBLIC  EGC_ON,EGC_OF

CODE    SEGMENT PUBLIC

        ASSUME CS:CODE

EGC_ON  PROC
        MOV     AL,7                    ;
        OUT     6AH,AL                  ;
        MOV     AL,5                    ;
        OUT     6AH,AL                  ;
        CLI                             ;
        MOV     AL,80H                  ;  EGC拡張モード・オン
        OUT     7CH,AL                  ;
        STI                             ;
        MOV     AL,6                    ;
        OUT     6AH,AL                  ;
        RET                             ;
EGC_ON  ENDP

EGC_OF  PROC
        MOV     AX,0FFF0H               ;
        MOV     DX,4A0H                 ;
        OUT     DX,AX                   ;
        MOV     AX,0FFFFH               ;
```

```
        MOV     DX,4A8H                    ;
        OUT     DX,AX                      ;
        MOV     AL,7                       ;
        OUT     6AH,AL                     ;
        MOV     AL,4                       ;} EGC拡張モード・オフ
        OUT     6AH,AL                     ;
        CLI                                ;
        XOR     AX,AX                      ;
        OUT     7CH,AL                     ;
        STI                                ;
        MOV     AL,6                       ;
        OUT     6AH,AL                     ;
        RET                                ;
EGC_OF  ENDP

CODE    ENDS

        END
```

これで、EGCの拡張モードのオン／オフが可能となりました。このEGC拡張モードの特徴は、プレーンに対する指定がないことです。つまり、G.VRAMアドレスの指定はCRT上の位置を決めるだけであり、実際のアクセスはB／R／G／Iいずれのプレーンでもかまいません。本書ではB面に統一していますが、もちろん他のプレーンでも差し支えありません。

では実際に、第1章にあった点を打つプログラム（LIST 1-1）と同じことを EGCを使って実現してみましょう。なお、ROPの入力として、フォアグラウンドカラーを選択した場合、ROPコードはBを使います。

```
A＞LIST2-2 ⏎
```

LIST 2-2

```
;*************************************
;*              LIST2-2              *
;*************************************

EXTRN   GSTAT:NEAR,GMD16:NEAR,EGC_ON:NEAR,EGC_OF:NEAR

CODE    SEGMENT PUBLIC

        ASSUME  CS:CODE,DS:CODE

OUTEGC  MACRO   PORT,DATA                  ;;ポート（PORT）へDATAをOUTする
        MOV     AX,DATA
        MOV     DX,PORT
        OUT     DX,AX
        ENDM
```

```
PMAIN:  CALL    GSTAT                   ;グラフィックスの開始
        JB      TEXIT                   ;異常終了であればTEXITへ
        CALL    GMD16                   ;640×400,16色モード・セット
        JB      TEXIT                   ;異常終了であればTEXITへ
        CALL    EGC_ON                  ;EGC拡張モード・オン
        OUTEGC  4A0H,0FFF0H             ;┐
        OUTEGC  4A2H,040FFH             ;│
        OUTEGC  4A4H,2CAAH              ;│
        OUTEGC  4A6H,15                 ;├ EGCの初期設定
        OUTEGC  4A8H,0FFFFH             ;│
        OUTEGC  4ACH,0                  ;│
        OUTEGC  4AEH,0                  ;┘
        PUSH    DS                      ;┐
        MOV     AX,BLUE                 ;│
        MOV     DS,AX                   ;│
        XOR     AX,AX                   ;├ 点を打つ
        MOV     DI,AX                   ;│
        DEC     AX                      ;│
        MOV     [DI],AX                 ;│
        POP     DS                      ;┘
        CALL    EGC_OF                  ;EGC拡張モード・オフ
TEXIT:  MOV     AX,4C00H                ;リターン・コード・セット
        INT     21H                     ;MS-DOSへ

CODE    ENDS

STACK   SEGMENT STACK                   ;スタック・セグメントの定義
        DW      100H    DUP(0)
STACK   ENDS

BLUE    SEGMENT AT 0A800H               ;B面用セグメントの定義
BLUE    ENDS

        END
```

LIST 2-2では実際にVRAMへのアクセスをしていますが、EGCのモードを設定している部分をカットすると、データをグラフィックVRAMに1回書き込んでいるだけです。これだけで、B/R/G/I各面に対する『ANDを取ってORを取る』という作業も完了です。そして、このときポート4A0H～4AEHへ出力するデータの内容がポイントなのです。

この設定で、特にわかりにくいのは、ポート4AEHの扱い方でしょう。これは、G.VRAMへ書き込むビット数を設定するためのポートですが、ビットの数え方が0から始まっています。例えば、点を打つ今回のプログラムでは1ビットの書き込みですから、設定するビット数は0となります。また、1バイト全てを埋める場合にはビット数は7となります。

各ポートへ出力するデータの内容を、図 2-2-2と照らし合わせながらよく確認してください。また、EGC の設定をいろいろと変化させて研究するのも役に立つことです。

2−3 重ね合わせ

PC-9801 シリーズの画面構成（図 1-1-1）から判断すると、グラフィック画面どうしの重ね合わせ処理など不可能なように見えます。しかし、それが可能であることはすでに多くのゲームによって証明されていますし、おそらく不可能と信じている人など1人もいないでしょう。それどころか、図 1-1-1以外に特殊なキャラクタ表示機能を持っているはずだと信じている人さえいるかもしれません。

残念なことに、たとえ EGC搭載機種であっても、グラフィックスに関してのハードウェア的な能力はこれまでに説明してきたことがすべてです。GDCという機能もありますが、基本的には変わりません。これ以外のことは、すべてプログラム・テクニックによってカバーされているのです。もちろん、最初から多様なテクニックが存在していたわけではありません。ハードウェアで用意されていない機能は、プログラマーが夢を追い求めて実現したものばかりです。つまり、PC-9801 シリーズというのは「テクニックを要求するが、テクニックの酷使にも耐えられる」ように設計された機種なのです。

画面処理テクニックへの第一歩、それはキャラクタの重ね合わせです。すでに基本的な原理はドット表示の際に示されています。これをキャラクタに拡大したのが次のプログラム（LIST 2-3）です。まずは、実行してみてください。

```
A>LIST2-3 ⏎
```

LIST 2-3

```
;*************************************
;*              LIST2-3              *
;*************************************

EXTRN   GSTAT:NEAR,GMD16:NEAR,TMEN1:BYTE,BMENS:NEAR

CODE    SEGMENT PUBLIC

        ASSUME  CS:CODE,DS:CODE

PMAIN:  CALL    GSTAT                       ;グラフィックスの開始
        JB      TEXIT                       ;異常終了であればTEXITへ
        CALL    GMD16                       ;640×400,16色モード・セット
        JB      TEXIT                       ;異常終了であればTEXITへ
        MOV     AX,BLUE                     ;┐
        MOV     DL,10011001B                ;│
        CALL    BACKD                       ;│
        MOV     AX,RED                      ;│
        MOV     DL,00110011B                ;│
        CALL    BACKD                       ;├ 背景の描写
        MOV     AX,GREEN                    ;│
        MOV     DL,11001100B                ;│
        CALL    BACKD                       ;│
        MOV     AX,ITSTY                    ;│
        MOV     DL,11111111B                ;│
```

```
        CALL    BACKD                       ;」
        MOV     SI,OFFSET BMENS             ;┐
        MOV     AX,BLUE                     ;│
        CALL    DICHR                       ;│
        MOV     AX,RED                      ;│
        CALL    DICHR                       ;├ おばけのピーヨの表示
        MOV     AX,GREEN                    ;│
        CALL    DICHR                       ;│
        MOV     AX,ITSTY                    ;│
        CALL    DICHR                       ;┘
TEXIT:  MOV     AX,4C00H                    ;リターン・コード・セット
        INT     21H                         ;MS-DOSへ戻る

BACKD   PROC
        PUSH    DS                          ;┐
        MOV     DS,AX                       ;│
        XOR     BX,BX                       ;│
        MOV     CL,200                      ;│
BACL0:  MOV     CH,80                       ;│
BACL1:  MOV     [BX],DL                     ;│
        MOV     BYTE PTR [BX+80],0          ;│
        ROL     DL,1                        ;│
        ROL     DL,1                        ;│
        INC     BX                          ;├ 背景の描写
        DEC     CH                          ;│
        JNE     BACL1                       ;│
        ADD     BX,80                       ;│
        ROL     DL,1                        ;│
        ROL     DL,1                        ;│
        DEC     CL                          ;│
        JNE     BACL0                       ;│
        POP     DS                          ;│
        RET                                 ;┘
BACKD   ENDP

DICHR   PROC
        PUSH    DS                          ;┐
        MOV     DS,AX                       ;│
        MOV     DI,OFFSET TMEN1             ;│
        MOV     BX,3C00H+26H                ;│
        MOV     CX,32                       ;│
DICL0:  MOV     AX,CS:[DI]                  ;│
        AND     [BX],AX                     ;│
        MOV     AX,CS:[SI]                  ;│
        OR      [BX],AX                     ;├ 透明データで画面をくり抜き(AND)
        MOV     AX,CS:[DI+2]                ;│ キャラクタを表示(OR)
        AND     [BX+2],AX                   ;│
        MOV     AX,CS:[SI+2]                ;│
        OR      [BX+2],AX                   ;│
        ADD     SI,4                        ;│
        ADD     DI,4                        ;│
        ADD     BX,80                       ;│
        LOOP    DICL0                       ;│
        POP     DS                          ;│
        RET                                 ;┘
DICHR   ENDP

CODE    ENDS

STACK   SEGMENT STACK                       ;スタック・セグメントの定義
        DW      100H    DUP(0)
STACK   ENDS
```

```
BLUE    SEGMENT AT 0A800H                  ;B面のセグメントの定義
BLUE    ENDS

RED     SEGMENT AT 0B000H                  ;R面のセグメントの定義
RED     ENDS

GREEN   SEGMENT AT 0B800H                  ;G面のセグメントの定義
GREEN   ENDS

ITSTY   SEGMENT AT 0E000H                  ;I 面のセグメントの定義
ITSTY   ENDS

        END
```

ピーヨデータ

```
;**************************************
;*             PIYO DATA              *
;**************************************

PUBLIC  TMEN1,BMENS,RMENS,GMENS,IMENS

CODE    SEGMENT PUBLIC

        ASSUME  CS:CODE

TMEN1   LABEL BYTE
        DB      0FFH,0F0H,   7,0FFH,0FFH,0E0H,   3,0FFH
        DB      0FFH,080H,   0,0FFH,0FFH,   0,   0, 7FH
        DB      0FEH,   0,   0, 3FH,0FCH,   0,   0, 1FH
        DB      0F8H,   0,   0, 0FH,0F8H,   0,   0, 0FH
        DB      0F0H,   0,   0,   7,0F0H,   0,   0,   7
        DB      0E0H,   0,   0,   3,0E0H,   0,   0,   3
        DB      0E0H,   0,   0,   1,0E0H,   0,   0,   1
        DB      0E0H,   0,   0,   1,0E0H,   0,   0,   1
        DB      0E0H,   0,   0,   1,0E0H,   0,   0,   1
        DB      0C0H,   0,   0,   1,0C0H,   0,   0,   1
        DB       80H,   0,   0,   3, 80H,   0,   0,   3
        DB         0,   0,   0,   5,   0,   0,   0,   5
        DB         0,   0,   0,   0,   0,   0,   0,   0
        DB       83H,   0,   0,   1,0C7H,0C0H,   0,   1
        DB      0FFH,0E0H,   0,   3,0FFH,0F8H,   0,   7
        DB      0FFH,0FEH,   0, 0FH,0FFH,0FFH, 80H, 3FH

BMENS   LABEL BYTE
        DB         0,   0,   0,   0,   0, 0AH,0A8H,   0
        DB         0, 1FH,0FCH,   0,   0, 7FH,0FEH,   0
        DB         0,0FFH,0FFH, 80H,   1,0FFH,0FFH,0C0H
        DB         3,0FFH,0FFH,0E0H,   3,0FFH,0FFH,0E0H
        DB         7, 9FH, 9FH,0F0H,   7, 0FH, 0FH,0F0H
        DB       0EH,   6,   7,0F8H, 0EH,   6,   7,0F8H
        DB       0EH,   6,   7,0FCH, 0EH,   6,   7,0FCH
        DB       0FH, 0FH, 0FH,0FCH, 0FH, 0FH, 0FH,0FCH
        DB       0FH,0FFH,0FFH,0FCH, 0FH,0FFH,0FFH,0FCH
        DB       1FH,0FFH,0FFH,0FCH, 1FH,0FFH,0FFH,0FCH
        DB       2BH,0BFH,0B2H, 78H, 2BH,0DFH, 72H, 78H
        DB       55H,0C0H,064H,0F0H, 55H,0E0H,0E4H,0F0H
        DB         0,0FFH,0E0H,0E2H,   0,0FFH,0E0H,0E6H
        DB         0, 3FH,0FFH,0CCH,   0, 1FH,0FFH, 9CH
        DB         0,   7,0FFH,0F8H,   0,   3,0FFH,0F0H
        DB         0,   0, 55H, 40H,   0,   0,   0,   0
```

```
RMENS    LABEL BYTE
         DB          0,   0,   0,   0,   0,   0,   0,   0
         DB          0,   2,0D0H,   0,   0,   6,0D0H,   0
         DB          0, 3FH,0FAH,   0,   0, 7FH,0FEH,   0
         DB          1, 9FH, 9EH, 80H,   1, 0FH, 0EH, 80H
         DB          2, 66H, 67H, 40H,   2,0F6H,0F7H, 40H
         DB          5,0F9H,0FBH,0A0H,   5,0F9H,0FBH,0A0H
         DB          5,0C9H, 3BH,0D0H,   5,0C9H, 3BH,0D0H
         DB          2,0F6H,0F7H,0A0H,   2,0F6H,0F7H,0A0H
         DB          7, 0FH, 0FH,0D0H,   7, 9FH, 9FH,0D0H
         DB        1BH,0FFH,0FFH,0A0H, 1BH,0FFH,0FFH,0A0H
         DB        39H,0BFH,0B7H, 20H, 39H,0DFH, 77H, 20H
         DB        7CH,0C0H,06EH,0C0H, 7CH, 60H,0EEH,0C0H
         DB        54H, 7FH,0EAH, 80H, 54H, 5FH,0EAH, 80H
         DB          0, 0AH,0FEH,   8,   0, 0AH,0FEH,   8
         DB          0,   0,08AH, 80H,   0,   0,0AAH,0A0H
         DB          0,   0,   0,   0,   0,   0,   0,   0

GMENS    LABEL BYTE
         DB          0,   0,   0,   0,   0, 0AH,0A8H,   0
         DB          0, 1FH,0FCH,   0,   0, 7FH,0FEH,   0
         DB          0,0FFH,0FFH, 80H,   1,0FFH,0FFH,0C0H
         DB          3, 9FH, 9FH,0E0H,   3, 0FH, 0FH,0E0H
         DB          6,0F6H,0F7H,0F0H,   6,0F6H,0F7H,0F0H
         DB        0DH,0F9H,0FBH,0F8H, 0DH,0F9H,0FBH,0F8H
         DB        0DH,0C9H, 3BH,0FCH, 0DH,0C9H, 3BH,0FCH
         DB        0EH,0F6H,0F7H,0FCH, 0EH,0F6H,0F7H,0FCH
         DB        0FH, 0FH, 0FH,0FCH, 0FH, 9FH, 9FH,0FCH
         DB        1FH,0FFH,0FFH,0FCH, 1FH,0FFH,0FFH,0FCH
         DB        3BH,0BFH,0B7H, 78H, 3BH,0DFH, 77H, 78H
         DB        7DH,0C0H,06EH,0F0H, 7DH,0E0H,0EEH,0F0H
         DB        54H,0FFH,0EAH,0E2H, 54H,0FFH,0EAH,0E6H
         DB          0, 3FH,0FFH,0CCH,   0, 1FH,0FFH, 9CH
         DB          0,   7,0FFH,0F8H,   0,   3,0FFH,0F0H
         DB          0,   0, 55H, 40H,   0,   0,   0,   0

IMENS    LABEL BYTE
         DB          0,   0,   0,   0,   0, 0AH,0A8H,   0
         DB          0, 1DH, 2CH,   0,   0, 79H, 2EH,   0
         DB          0,0C0H,   5, 80H,   1, 80H,   1,0C0H
         DB          2,   0,   1, 60H,   2,   0,   1, 60H
         DB          4, 90H, 90H,0B0H,   4,   0,   0,0B0H
         DB          8,   0,   0, 58H,   8,   0,   0, 58H
         DB          8,   0,   0, 2CH,   8,   0,   0, 2CH
         DB        0CH,   0,   0, 5CH, 0CH,   0,   0, 5CH
         DB          8,   0,   0, 2CH,   8,   0,   0, 2CH
         DB          4,   0,   0, 5CH,   4,   0,   0, 5CH
         DB          2,   0,   0, 58H,   2,   0,   0, 58H
         DB          1,   0,   0, 30H,   1, 80H,   0, 30H
         DB          0, 80H,   0, 62H,   0,0A0H,   0, 66H
         DB          0, 35H,   1,0C4H,   0, 15H,   1, 94H
         DB          0,   7, 75H, 78H,   0,   3, 55H, 50H
         DB          0,   0, 55H, 40H,   0,   0,   0,   0

CODE     ENDS

         END
```

色盲検査表のような背景模様の中央に、本書のマスコット・キャラクタである『おばけのピーヨ』が現れました。プログラムは、例の『AND を取ってORを取る』という重ね合わせの基本そのものです。キャラクタ・データは透明（くり抜き）＋B面＋R面＋G面＋I面の順で並んでいます。今回の『おばけのピーヨ』データは、この先いろいろなプログラムで利用しますので、外部モジュールとしてあります。

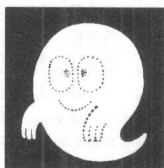
最初に透明データによってキャラクタ部分をくり抜いていますが、このときの透明データ構造は図 2-3-1のようになっています。

黒い部分のビット=1（背景が残る）
白い部分のビット=0（くり抜かれる）

図 2-3-1　透明データの構造

この例ではEGCを使わずに単純にG.VRAMへ4回アクセスしています。では、次に同じことを EGCを利用してやってみましょう。といっても、まったく同じではツマラナイので、せめてもの工夫で背景をGRCGのTDWモードを利用して描き、模様も変えてみました。

```
A>LIST2-4⏎
```

LIST 2-4

```
;*********************************
;*            LIST2-4            *
;*********************************

EXTRN   GSTAT:NEAR ,GMD16:NEAR,EGC_ON:NEAR,EGC_OF:NEAR,TMEN1:BYTE

CODE    SEGMENT PUBLIC

        ASSUME  CS:CODE

GRCGSET MACRO   PLB,PLR,PLG,PLI,DATA      ;;GRCG設定用マクロ定義
        CLI
        MOV     AL,080H
        OUT     7CH,AL
        STI
        MOV     AL,PLB
        OUT     7EH,AL
        MOV     AL,PLR
        OUT     7EH,AL
        MOV     AL,PLG
        OUT     7EH,AL
        MOV     AL,PLI
        OUT     7EH,AL
        MOV     DX,DATA
        ENDM

OUTEGC  MACRO   PORT,DATA                 ;;ポート（PORT）へDATAをOUTする
        MOV     AX,DATA
        MOV     DX,PORT
        OUT     DX,AX
        ENDM

PMAIN:  CLD
        CALL    GSTAT                     ;グラフィックスの開始
        JNB     $+5                       ;
        JMP     TEXIT                     ;異常終了であればTEXITへ
        CALL    GMD16                     ;640×400,16色モード・セット
        JNB     $+5                       ;
        JMP     TEXIT                     ;異常終了であればTEXITへ
        MOV     AX,BLUE                   ;
        MOV     ES,AX                     ;
        GRCGSET 0,0,0AAH,0FFH,0           ;  GRCGによる背景の描写
        CALL    BACKD                     ;
        GRCGSET 0,0FFH,55H,0FFH,1         ;
        CALL    BACKD                     ;
        CALL    EGC_ON                    ;EGC拡張モード・オン
        OUTEGC  4A2H,0FFH                 ;
        OUTEGC  4A4H,0CC0H                ;
        OUTEGC  4A8H,0FFFFH               ;  EGCの初期設定
        OUTEGC  4ACH,0                    ;
        OUTEGC  4AEH,<8*4-1>              ;
        MOV     SI,OFFSET TMEN1           ;
        MOV     AX,0FFF0H                 ;
        MOV     DX,4A0H                   ;
        CALL    DICHR                     ;
        MOV     AX,0CFCH                  ;
        MOV     DX,4A4H                   ;  ピーヨの表示
        OUT     DX,AX                     ;
        MOV     AX,0FFFEH                 ;
        MOV     DX,4A0H                   ;
        CALL    DICHR                     ;
        CALL    DICHR                     ;
        CALL    DICHR                     ;
```

```
        CALL    DICHR                   ;J
        CALL    EGC_OF                  ;EGC拡張モード・オフ
TEXIT:  MOV     AX,4C00H                ;リターン・コード・セット
        INT     21H                     ;MS-DOSへ

BACKD   PROC
        MOV     AH,400/8                ;
BACL0:  MOV     CL,40                   ;
BACL1:  MOV     CH,8                    ;
        MOV     DI,DX                   ;
BACL2:  STOSB                           ;
        ADD     DI,79                   ;
        DEC     CH                      ;
        JNE     BACL2                   ; 背景の描写
        ADD     DX,2                    ;
        DEC     CL                      ;
        JNE     BACL1                   ;
        ADD     DX,80*7                 ;
        XOR     DX,1                    ;
        DEC     AH                      ;
        JNE     BACL0                   ;
        RET                             ;
BACKD   ENDP

DICHR   PROC
        OUT     DX,AX                   ;
        MOV     DI,3C00H+26H            ;
        MOV     CX,32                   ;
DICLP:  MOVSW                           ;
        MOVSW                           ; ピーヨの表示
        ADD     DI,80-4                 ;
        LOOP    DICLP                   ;
        ROL     AX,1                    ;
        RET                             ;
DICHR   ENDP

CODE    ENDS

STACK   SEGMENT STACK                   ;スタック・セグメントの定義
        DW      100H    DUP(0)
STACK   ENDS

BLUE    SEGMENT AT 0A800H               ;B面用セグメントの定義
BLUE    ENDS

        END
```

透明データで、一度に4画面に対してANDを取っています。4画面同時アクセスですから、アクティブプレーンPB〜PIは0、すなわちポート4A0Hへ出力するデータは FFF0Hとなります。ROPコードは、CPUデータとG.VRAM上のデータとのANDですから、(S・D) でC0Hです。参考までに、PC-8801SR以降のALU用に作られた透明データを、PC-9801のEGCの拡張モードでそのまま利用する場合は、反転したCPUのデータとのANDですから 0CHです。くり抜きが完了したら、データをB面→R面→G面→I面と順番にOR (ROPコードFCH) 出力するだけOKです。

ところで、ダメージを受けたときなどによくカラー反転表示が使われますが、これはROPのビット反転 (XOR) 機能を利用すれば簡単にできます。XORによるビッ

ト反転は次のような演算ですが、元データのビットが0ならビットセット（ORを取る）と結果は同じです。

元データ：	00000000B	11111111B
データ：	XOR 11100111B	XOR 11100111B
新データ：	11100111B	00011000B

したがって、重ね合わせたキャラクタが反転カラーになるためには、透明データでくり抜いた部分を上の演算のように逆に埋めなければなりません。すなわち、「くり抜く／埋める」を交互に行えば「ノーマル表示／反転カラー表示」を繰り返すことになり、ダメージなどの表現が簡単にできるわけです。LIST 2-4は、スペースキーを押すたびに表示が変わるよう、透明データ使用時のデータをそのままアンドを取るか、反転後にオアを取るか、ROPコードを C0H／CFHと切り換えています。この切り換えにも XORを利用しています（TLOOP の行）。また、B／R／G／I 各プレーンに対するROPコードは（D (+)S）ですから3CHとなります。

```
A>LIST2-5 ⏎
```

LIST 2-5

```
;*************************************
;*            LIST2-5               *
;*************************************

EXTRN   GSTAT:NEAR,GMD16:NEAR,EGC_ON:NEAR,EGC_OF:NEAR,TMEN1:BYTE

CODE    SEGMENT PUBLIC

        ASSUME  CS:CODE,DS:CODE

GRCGSET MACRO   PLB,PLR,PLG,PLI,DATA       ;;GRCGの設定用マクロ定義
        CLI
        MOV     AL,080H
        OUT     7CH,AL
        STI
        MOV     AL,PLB
        OUT     7EH,AL
        MOV     AL,PLR
        OUT     7EH,AL
        MOV     AL,PLG
        OUT     7EH,AL
        MOV     AL,PLI
        OUT     7EH,AL
        MOV     DX,DATA
        ENDM

OUTEGC  MACRO   PORT,DATA                  ;;ポート（PORT）へDATAをOUTする
        MOV     AX,DATA
        MOV     DX,PORT
        OUT     DX,AX
        ENDM

GETKEY  MACRO                              ;;1文字入力用マクロ定義
        LOCAL   GLOOP
GLOOP:  MOV     DL,0FFH
```

```
        MOV     AH,6
        INT     21H
        JZ      GLOOP
        ENDM

PRINT   MACRO   STRING                          ;;文字列出力用マクロ定義
        MOV     DX,OFFSET STRING
        MOV     AH,09
        INT     21H
        ENDM

ESCKY   EQU     1BH                             ; [ESC] キーのアスキーコード

PMAIN:  CALL    GSTAT                           ;グラフィックスの開始
        JNB     $+5                             ;
        JMP     TEXIT                           ;異常終了であればTEXITへ
        CALL    GMD16                           ;640×400, 16色モード・セット
        JNB     $+5                             ;
        JMP     TEXIT                           ;異常終了であればTEXITへ
        PRINT   TXCLR                           ;テキスト画面クリア
        MOV     AX,BLUE                         ;
        MOV     ES,AX                           ;
        GRCGSET 0,0,0AAH,0FFH,0                 ;  GRCGによる背景の描写
        CALL    BACKD                           ;
        GRCGSET 0,0FFH,55H,0FFH,1               ;
        CALL    BACKD                           ;
        CALL    EGC_ON                          ;EGC拡張モード・オン
        OUTEGC  4A2H,0FFH                       ;
        OUTEGC  4A8H,0FFFFH                     ;  EGCの初期設定
        OUTEGC  4ACH,0                          ;
        OUTEGC  4AEH,<8*4-1>                    ;
        CLD
TLOOP:  XOR     BYTE PTR XORSET,0FH             ;
        MOV     AL,XORSET                       ;
        MOV     AH,0CH                          ;
        MOV     DX,4A4H                         ;
        OUT     DX,AX                           ;
        MOV     SI,OFFSET TMEN1                 ;
        MOV     AX,0FFF0H                       ;
        CALL    DICHR                           ;  表示(ノーマル/反転)を交互に行う
        OUTEGC  4A4H,0C3CH                      ;
        MOV     AX,0FFFEH                       ;
        CALL    DICHR                           ;
        CALL    DICHR                           ;
        CALL    DICHR                           ;
        CALL    DICHR                           ;
KEYIN:  GETKEY                                  ;
        CMP     AL,ESCKY                        ;  [ESC] キーが押されていればEGCOFへ
        JE      EGCOF                           ;
        CMP     AL," "                          ;
        JNE     KEYIN                           ;  [SPACE] が押されていればTLOOPへ
        JMP     TLOOP                           ;
EGCOF:  CALL    EGC_OF                          ;EGC拡張モード・オフ
        PRINT   TKEEP                           ;
TEXIT:  MOV     AX,4C00H                        ;  MS-DOSへ
        INT     21H                             ;

XORSET  DB      0CFH                            ;XORデータ

BACKD   PROC
        MOV     AH,400/8                        ;
BACL0:  MOV     CL,40                           ;
BACL1:  MOV     CH,8                            ;
        MOV     DI,DX                           ;
```

```
BACL2:  STOSB                           ;
        ADD     DI,79                   ;
        DEC     CH                      ;  背景の表示
        JNE     BACL2                   ;
        ADD     DX,2                    ;
        DEC     CL                      ;
        JNE     BACL1                   ;
        ADD     DX,80*7                 ;
        XOR     DX,1                    ;
        DEC     AH                      ;
        JNE     BACL0                   ;
        RET                             ;
BACKD   ENDP

DICHR   PROC
        MOV     DX,04A0H                ;
        OUT     DX,AX                   ;
        MOV     DI,CS:PIYOX             ;
        MOV     CX,32                   ;
DICLP:  MOVSW                           ;  ピーヨの表示
        MOVSW                           ;
        ADD     DI,80-4                 ;
        LOOP    DICLP                   ;
        ROL     AX,1                    ;
        RET                             ;
DICHR   ENDP

PIYOX   DW      3C00H+26H               ;ピーヨの座標

TXCLR   LABEL BYTE                      ;テキスト・クリア&文字の属性の保存
        DB      1BH,"[2J"
        DB      1BH,"[s",1BH,"[>5h",1BH,"[1>h$"

TKEEP   LABEL BYTE                      ;文字の属性の復元
        DB      1BH,"[u",1BH,"[>5l",1BH,"[1>l$"

CODE    ENDS

STACK   SEGMENT STACK                   ”スタック・セグメントの定義
        DW      100H    DUP(0)
STACK   ENDS

BLUE    SEGMENT AT 0A800H               ”B面用セグメントの定義
BLUE    ENDS

        END
```

このプログラムでは、反転カラー時にくり抜き部を4面とも埋めていますが、これを面によって変えるとまた違った反転カラーキャラクタを作ることができます。変更の方法は簡単で、B／R／G／I面アクセス時のポート4A0Hのアクティブプレーン設定の組み合わせを変えるだけです。

このあたりの変化は、文章で1つひとつ説明するとキリがありませんから、いろいろと適当に数値を変化させながら実験してください。それによって、EGCのモードの違いや利用法が自然と理解できるようになるでしょう。

2−4　4面同時リード／ライト

簡単なことでも、同じことを4度繰り返すのは面倒なものです。もしも言葉を正確にするという妙な理由で、『今日からすべての文字は4度繰り返すこと』などという法律ができとしたら……。想像しただけでも大変です。

「こここここんんんんにににちちちちわわわわ」

「ややややああああ、、、、どどどどううううもももも」

こんなバカなことが現実に起こるとは思えませんが、ここコンピュータ98国のグラフィックVRAM地方では、それに近いことをしなければなりません。画面を作成するときは、各プレーンにバラバラの値を入れるわけですから、同じような作業が続いてもある程度は仕方のないことと納得できます。しかし、一旦画面に絵として描かれたなら、プレーンという感覚はなくなるのが普通です。例えば、グラフィック画面をロールアップしたりダウンしたりする場合、プレーンごとに同じ作業（データの転送）を4度繰り返すというのは当然とはいえ面倒なことです。

こんなとき便利なのが、拡張モードの『直前のリードサイクルでリードされたデータを4面同時にライトする』というモードです（図 2-2-2)。これは、4画面分のデータを一括して転送する機能です。しかも、ビット単位のシフト機能がありますから、アイデア次第で利用法は大きく広がります。

参考例として、この4面同時転送機能と、ビット単位のシフト機能を利用したグラフィック画面をドット単位に縦横自在に移動させるプログラム（LIST 2-6）を紹介します。これは、テンキー（[2],[4],[6],[8]）の操作で画面を上下左右にスクロール・ループさせるものです。

なお、実行前にグラフィック画面に何か描いておかないと、闇夜のカラスになってしまいます。

LIST 2-6

```
;************************************
;*            LIST2-6               *
;************************************

EXTRN   GSTAT:NEAR,GMD16:NEAR,EGC_ON:NEAR,EGC_OF:NEAR

CODE    SEGMENT PUBLIC

        ASSUME  CS:CODE,DS:CODE

OUTEGC  MACRO   PORT,DATA                    ;;指定ポート（PORT）へDATAをOUTする
        MOV     AX,DATA
        MOV     DX,PORT
        OUT     DX,AX
        ENDM

GETKEY  MACRO                                ;;1文字入力
        LOCAL   GLOOP
```

```
GLOOP:  MOV     DL,0FFH
        MOV     AH,6
        INT     21H
        JZ      GLOOP
        ENDM

PRINT   MACRO   STRING                     ;;文字列出力
        MOV     DX,OFFSET STRING
        MOV     AH,09
        INT     21H
        ENDM

ESCKY   EQU     1BH                        ; [ESC] キーのアスキーコード

PMAIN:  CALL    GSTAT                      ;グラフィックスの開始
        JB      TEXIT                      ;異常終了であればTEXITへ
        CALL    GMD16                      ;640×400,16色モード・セット
        JB      TEXIT                      ;異常終了であればTEXITへ
        PRINT   TXCLR                      ;テキスト画面クリア
        MOV     AX,CS                      ;
        MOV     ES,AX                      ;ES←CS
        CALL    EGC_ON                     ;EGC拡張モード・オン
        OUTEGC  4A0H,0FFF0H                ;┐
        OUTEGC  4A2H,0FFH                  ;│ EGCの初期設定
        OUTEGC  4A4H,28F0H                 ;│
        OUTEGC  4A8H,0FFFFH                ;┘
TLOOP:  GETKEY                             ;1文字入力
        MOV     CX,4                       ;┐
        MOV     DI,OFFSET KEYTBL           ;│
        CLD                                ;│
        REPNZ   SCASB                      ;│
        JNE     TEST1                      ;│
        MOV     BX,OFFSET ADRTBL           ;│ [2][4][6][8]に従ってそれぞれの
        ADD     BX,CX                      ;│ 処理をする
        ADD     BX,CX                      ;│
        CALL    [BX]                       ;│
        JMP     TLOOP                      ;┘
TEST1:  CMP     AL,ESCKY                   ;┐
        JE      EGCOF                      ;│ [ESC] キーが押されていなければTLOOPへ
        JMP     TLOOP                      ;┘
EGCOF:  CALL    EGC_OF                     ;EGC拡張モード・オフ
        PRINT   TKEEP                      ;┐
TEXIT:  MOV     AX,4C00H                   ;│ MS-DOSへ
        INT     21H                        ;┘

MOVE2   PROC                               ;┐
        PUSH    DS                         ;│
        PUSH    ES                         ;│
        MOV     AX,BLUE                    ;│
        MOV     DS,AX                      ;│
        MOV     ES,AX                      ;│
        MOV     SI,7CFEH                   ;│
        MOV     DI,7CFEH+80                ;│
        MOV     CX,40*400                  ;│
        STD                                ;│ ドット単位でグラフィック画面を下へ
        OUTEGC  4ACH,1000H                 ;│ スクロール
        OUTEGC  4AEH,639                   ;│
        REP     MOVSW                      ;│
        MOV     SI,07CFEH+80               ;│
        MOV     CX,40                      ;│
        REP     MOVSW                      ;│
        POP     ES                         ;│
        POP     DS                         ;┘
```

```
        RET                                   ;┘
MOVE2   ENDP

MOVE4   PROC
        PUSH      DS                          ;┐
        PUSH      ES                          ;│
        MOV       AX,BLUE                     ;│
        MOV       DS,AX                       ;│
        MOV       ES,AX                       ;│
        XOR       SI,SI                       ;│
        MOV       DI,SI                       ;│
        MOV       BP,400                      ;│
        MOV       BX,7D00H                    ;│
        CLD                                   ;│
MV4LP:  OUTEGC    4ACH,0                      ;│
        OUTEGC    4AEH,0FH                    ;│
        MOV       AX,[SI]                     ;│
        MOV       [BX],AX                     ;│
        OUTEGC    4ACH,1                      ;├ ドット単位でグラフィック画面を左へ
        OUTEGC    4AEH,639                    ;│ スクロール
        MOV       AX,[SI]                     ;│
        MOV       [DI],AX                     ;│
        ADD       SI,2                        ;│
        MOV       CX,40-1                     ;│
        REP       MOVSW                       ;│
        MOV       AX,[BX]                     ;│
        MOV       [DI],AX                     ;│
        ADD       DI,2                        ;│
        DEC       BP                          ;│
        JNE       MV4LP                       ;│
        POP       ES                          ;│
        POP       DS                          ;│
        RET                                   ;┘
MOVE4   ENDP

MOVE6   PROC
        PUSH      DS                          ;┐
        PUSH      ES                          ;│
        MOV       AX,BLUE                     ;│
        MOV       DS,AX                       ;│
        MOV       ES,AX                       ;│
        MOV       SI,7CFEH                    ;│
        MOV       DI,7CFEH                    ;│
        MOV       BP,400                      ;│
        MOV       BX,7D00H                    ;│
        STD                                   ;│
MV6LP:  OUTEGC    4ACH,0                      ;│
        OUTEGC    4AEH,0FH                    ;│
        MOV       AX,[SI]                     ;│
        MOV       [BX],AX                     ;│
        OUTEGC    4ACH,1001H                  ;├ ドット単位でグラフィック画面を右へ
        OUTEGC    4AEH,639                    ;│ スクロール
        MOV       AX,[SI]                     ;│
        MOV       [DI],AX                     ;│
        SUB       SI,2                        ;│
        MOV       CX,40-1                     ;│
        REP       MOVSW                       ;│
        MOV       AX,[BX]                     ;│
        MOV       [DI],AX                     ;│
        SUB       DI,2                        ;│
        DEC       BP                          ;│
        JNE       MV6LP                       ;│
        POP       ES                          ;│
```

```
        POP     DS                      ;
        RET                             ;
MOVE6   ENDP

MOVE8   PROC
        PUSH    DS                      ;
        PUSH    ES                      ;
        MOV     AX,BLUE                 ;
        MOV     DS,AX                   ;
        MOV     ES,AX                   ;
        XOR     SI,SI                   ;
        MOV     DI,7D00H                ;
        OUTEGC  4ACH,0                  ;
        OUTEGC  4AEH,639                ; ドット単位でグラフィック画面を上へ
        CLD                             ; スクロール
        MOV     CX,40                   ;
        REP     MOVSW                   ;
        XOR     DI,DI                   ;
        MOV     CX,40*400               ;
        REP     MOVSW                   ;
        POP     ES                      ;
        POP     DS                      ;
        RET                             ;
MOVE8   ENDP

KEYTBL  LABEL BYTE                      ;キー・コード・テーブル
        DB      "2","4","6","8"

ADRTBL  LABEL WORD                      ;キー別の処理のベクタ・テーブル
        DW      MOVE8,MOVE6,MOVE4,MOVE2

TXCLR   LABEL BYTE                      ;テキスト・クリア　&　文字の属性の保存
        DB      1BH,"[2J"
        DB      1BH,"[s",1BH,"[>5h",1BH,"[1>h$"

TKEEP   LABEL BYTE                      ;文字の属性の復元
        DB      1BH,"[u",1BH,"[>5l",1BH,"[1>l$"

CODE    ENDS

STACK   SEGMENT STACK                   ;スタック・セグメントの定義
        DW      100H    DUP(0)
STACK   ENDS

BLUE    SEGMENT AT 0A800H               ;B面用セグメントの定義
BLUE    ENDS

        END
```

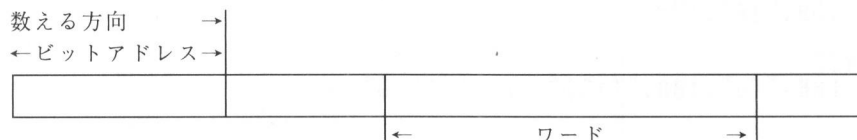
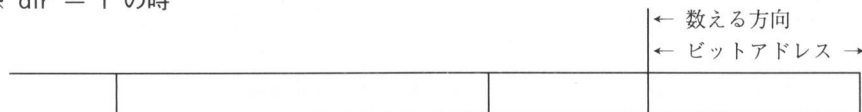
LIST 2-6では、ビット単位のシフト機能を使って、横方向を1ドット毎にスクロールさせています。メモリの基本単位が8ビット（EGCは16ビットがアクセス単位）であることを考慮すると、これは画期的なことです。スクロールのための領域転送にしても、単純にドットを打つ処理にしても、ポートの設定をしてVRAMをアクセスするという基本プロセスは従来と同じです。ここで、言及しなければならないのは、ポートの4ACHです。

実は、ポート4ACHはビット単位のシフトに関するポートなのです。このポートには、今まで0を設定してきましたが、それにはそれなりの意味があったからです。もう一度、図 2-2-2のポート4ACHの項を参照してください。

データの転送方向、転送元の開始ビットアドレス、転送先の開始ビットアドレスが設定できるようになっています。ビット12は転送方向を示しています（本書ではdirビットと略記）。アドレスの低い方から高い方へ転送する場合が0、逆に高い方から低い方へ転送する場合が1です。このような使い分けは、ストリング命令におけるディレクション・フラグと同じです。

dir ＝ 0 ･･･ アドレスの低い方から高い方へ転送
dir ＝ 1 ･･･ アドレスの高い方から低い方へ転送

転送方向を決めたら、ワード内のビットアドレスを設定します。ビット0～3が転送元のビットアドレスで、ビット4～7が転送先のビットアドレスです。各ビットアドレスは、dirビットによって数える方向が異なります。

※ dir ＝ 0 の時

※ dir ＝ 1 の時

転送ワード数はビット数をワード境界で切り上げた値を使います。もし両端で切り上げが生じると、＋2の切り上げとなります。ポート4ACHに関しては、LIST 2-6を参照しながらいろいろと操作してみてください。

2－5　グラフィックVRAMの余り

日本人は余暇の利用が下手といわれていますが、中には遊ぶために遊び方を勉強するという人もいるそうです。また、それがイヤなために「無理に遊ぶくらいなら仕事をするほうがマシ!!」と言う人もいます。一番いいのは、遊びと仕事が同じになることですが、せっかくの遊びも仕事となると楽しんでばかりはいられなくなるようです。コンピュータなどその典型かもしれません……。

余暇の利用というわけではありませんが、前節のプログラム（LIST 2-6）ではグラフィックVRAMの 各プレーンのオフセットアドレスの7D00H以降を、グラフィック・データの一時退避用として大いに活用していました。このグラフィックVRAMの オフセットアドレスの7D00H以降というのは、雰囲気的には単なる余りのような存在ですが、実は大変利用価値の高いメモリエリアなのです。というのは、同じアドレスに4画面分のデータを入れることができる上、その内容が画面に表示されないという特徴があるからです。

例えば、重ね合わせ処理をしたキャラクタ・パターンを動かす場合、実際には「背景でキャラクタを消去し、その上に新たなパターンを重ね合わせて表示する」という作業が必要です。

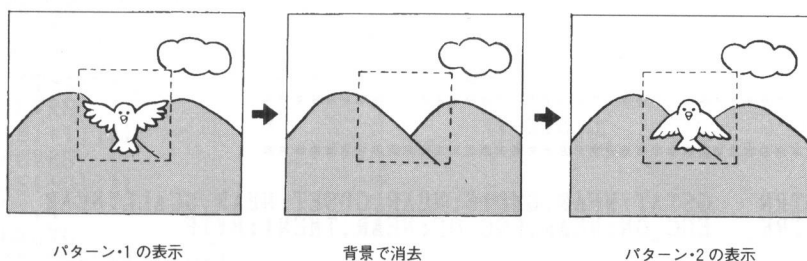



図 2-5-1　重ね合わせ処理とアニメーションの原理

これをまともに実行すれば、背景でキャラクタを消去した時点で画面はキャラクタ不在の状態になります。もちろん、その直後には新たなキャラクタ・パターンが描かれるわけですが、背景データの構造／格納法、キャラクタのサイズ……等、条件によってはキャラクタ不在という事実がチラツキとなって現れてくることがあります。これを解決するためには、キャラクタの消去（背景の表示）と新キャラクタの表示を同時に行わなければなりません。つまり、『背景表示＋くり抜き（AND）＋キャラクタ（OR）』というデータ処理を、グラフィックVRAMにデータを入れる前に行うということです。

これはけっして不可能なことではありませんが、1つのキャラクタに対し背景パターンが複数で構成されている場合（ゲームではこのようなケースも多い）、レジスタが不足してプログラムは難解複雑になります。そこで、余っているグラフィックVRAMの オフセットアドレスの7D00H以降を利用しようというのです。手順としては、図 2-5-2のようなものになります。

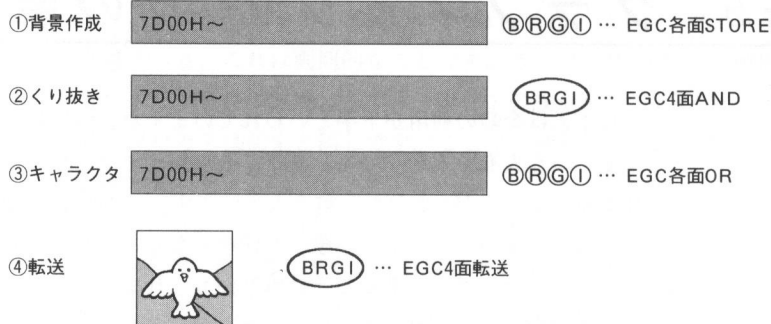



図 2-5-2　グラフィックVRAMの7D00H以降の利用手順

7D00H 以降への表示（実際には見えない）は、アドレスが連続しているので非常に簡単です（単なるデータの連続転送でよい）。次に、それを EGCの 4 面同時転送機能を使って表示アドレスへ転送すれば、消去と表示の同時処理が実現できることになります。

では、これら 3 種類の「重ね合わせアニメ処理」を『おばけのピーヨ』を使って実験してみましょう。プログラム（LIST 2-7）を実行してください。

```
A>LIST2-7⏎
```

LIST2-7

```
;*************************************
;*              LIST2-7              *
;*************************************

EXTRN   GSTAT:NEAR,GMD16:NEAR,GDSET:NEAR,GCALL:NEAR
EXTRN   EGC_ON:NEAR,EGC_OF:NEAR,TMEN1:BYTE

CODE    SEGMENT PUBLIC

        ASSUME  CS:CODE,DS:CODE

GRCGSET MACRO   PLB,PLR,PLG,PLI,DATA      ;;グラフィック・チャージャのレジスタセット用
        CLI
        MOV     AL,080H
        OUT     7CH,AL
        STI
        MOV     AL,PLB
        OUT     7EH,AL
        MOV     AL,PLR
        OUT     7EH,AL
        MOV     AL,PLG
        OUT     7EH,AL
        MOV     AL,PLI
        OUT     7EH,AL
        MOV     DX,DATA
        ENDM

OUTEGC  MACRO   PORT,DATA                 ;;指定ポート（DX）へDATAをOUTする
        MOV     AX,DATA
        MOV     DX,PORT
```

```
        OUT     DX,AX
        ENDM

GETKEY  MACRO                                  ;;1文字入力
        LOCAL   GLOOP
GLOOP:  MOV     DL,0FFH
        MOV     AH,6
        INT     21H
        JZ      GLOOP
        ENDM

PRINT   MACRO   STRING                         ;;文字列出力
        MOV     DX,OFFSET STRING
        MOV     AH,09
        INT     21H
        ENDM

ESCKY   EQU     1BH                            ; ESC キーのアスキーコード

PMAIN:  CALL    GSTAT                          ;グラフィックスの開始
        JNB     $+5                            ;
        JMP     TEXIT                          ;異常終了であればTEXITへ
        CALL    GMD16                          ;640×400,16色モード・セット
        JNB     $+5                            ;
        JMP     TEXIT                          ;異常終了であればTEXITへ
        PRINT   TXCLR                          ;テキスト画面クリア
        MOV     BX,OFFSET TMEN1                ;
        MOV     SI,OFFSET RVDAT                ;
        MOV     CX,32*5                        ;
DREVL:  CALL    REVRS                          ;
        MOV     [SI+3],AL                      ;
        CALL    REVRS                          ;
        MOV     [SI+2],AL                      ;  ピーヨ反転データ作成
        CALL    REVRS                          ;
        MOV     [SI+1],AL                      ;
        CALL    REVRS                          ;
        MOV     [SI+0],AL                      ;
        ADD     SI,4                           ;
        LOOP    DREVL                          ;
        MOV     AX,OFFSET RVDAT                ;
        XOR     AX,OFFSET TMEN1                ;
        MOV     SWITCH,AX                      ;
        MOV     AX,BLUE                        ;
        MOV     ES,AX                          ;  背景を表示
        GRCGSET 0,0,0AAH,0FFH,0                ;
        CALL    BACKD                          ;
        GRCGSET 0,0FFH,55H,0FFH,1              ;
        CALL    BACKD                          ;
        CLI                                    ;
        XOR     AL,AL                          ;
        OUT     7CH,AL                         ;
        STI                                    ;
        MOV     AX,CS                          ;
        MOV     ES,AX                          ;
        MOV     DI,OFFSET BKDAT                ;
        MOV     AX,BLUE                        ;
        CALL    BMAKE                          ;
        MOV     AX,RED                         ;
        CALL    BMAKE                          ;  ピーヨ表示（方法-3）
        MOV     AX,GREEN                       ;
        CALL    BMAKE                          ;
        MOV     AX,ITSTY                       ;
        CALL    BMAKE                          ;
```

```
        MOV     AX,BLUE             ;
        MOV     ES,AX               ;ES←BLUE
DLOOP:  CALL    PUTP1               ;ピーヨ表示（方法-1)
        CALL    PUTP2               ;ピーヨ表示（方法-2)
        CALL    PUTP3               ;ピーヨ表示（方法-3)
        MOV     AX,SWITCH           ;
        XOR     PATAD,AX            ;パターン番号（1／0）を変更する
        MOV     CH,10               ;
VWAIT:  IN      AL,0A0H             ;
        TEST    AL,00100000B        ;
        JNE     VWAIT               ;
VWAT2:  IN      AL,0A0H             ;  ウェイト
        TEST    AL,00100000B        ;
        JE      VWAT2               ;
        DEC     CH                  ;
        JNE     VWAIT               ;
KSLP1:  GETKEY                      ;1文字入力
        CMP     AL,ESCKY            ; [ESC] キーが押されていればKYESCへ
        JE      KYESC               ;
        CMP     AL," "              ;  [SPACE] キーが押されていなければ
        JNE     KSLP1               ;  KSLP1へ
        JMP     DLOOP               ;
KYESC:  PRINT   TKEEP               ;
TEXIT:  MOV     AX,4C00H            ;  MS-DOSへ
        INT     21H                 ;

SWITCH  DW      0                   ;パターン番号
PATAD   DW      TMEN1               ;パターン・アドレス

REVRS   PROC
        PUSH    CX                  ;
        MOV     CX,8                ;
        MOV     AH,[BX]             ;
REVL1:  RCR     AH,1                ;
        RCL     AL,1                ;  左右反転したデータを求める
        LOOP    REVL1               ;
        INC     BX                  ;
        POP     CX                  ;
        RET                         ;
REVRS   ENDP

BMAKE   PROC
        PUSH    DS                  ;
        MOV     DS,AX               ;
        XOR     SI,SI               ;
        MOV     AX,32               ;
        CLD                         ;
BMAKL:  MOVSW                       ;  プレーン別にデータを転送する
        MOVSW                       ;
        ADD     SI,80-4             ;
        DEC     AX                  ;
        JNE     BMAKL               ;
        POP     DS                  ;
        RET                         ;
BMAKE   ENDP

BACKD   PROC
        MOV     AH,400/8            ;
BACL0:  MOV     CL,40               ;
BACL1:  MOV     CH,8                ;
        MOV     DI,DX               ;
BACL2:  STOSB                       ;
        ADD     DI,79               ;
```

```
        DEC     CH                  ;┐
        JNE     BACL2               ;│ 背景を表示する
        ADD     DX,2                ;│
        DEC     CL                  ;│
        JNE     BACL1               ;│
        ADD     DX,80*7             ;│
        XOR     DX,1                ;│
        DEC     AH                  ;│
        JNE     BACL0               ;│
        RET                         ;┘
BACKD   ENDP

PUTP1   PROC
        CALL    EGC_ON              ;┐
        OUTEGC  4A2H,0FFH           ;│
        OUTEGC  4A4H,0CF0H          ;│
        OUTEGC  4A8H,0FFFFH         ;│
        OUTEGC  4ACH,0              ;│
        OUTEGC  4AEH,<8*4-1>        ;│
        MOV     SI,OFFSET BKDAT     ;│
        MOV     AX,0FFFEH           ;│
        CALL    PUT1D               ;│
        CALL    PUT1D               ;│
        CALL    PUT1D               ;│ EGCを利用してピーヨを指定位置に
        CALL    PUT1D               ;│ 表示
        OUTEGC  4A4H,0CC0H          ;│
        MOV     SI,PATAD            ;│
        MOV     AX,0FFF0H           ;│
        CALL    PUT1D               ;│
        OUTEGC  4A4H,0CFCH          ;│
        MOV     AX,0FFFEH           ;│
        CALL    PUT1D               ;│
        CALL    PUT1D               ;│
        CALL    PUT1D               ;│
        CALL    PUT1D               ;│
        CALL    EGC_OF              ;│
        RET                         ;┘
PUTP1   ENDP

PUT1D   PROC
        MOV     DX,4A0H             ;┐
        OUT     DX,AX               ;│
        MOV     DI,3C00H+1EH        ;│
        MOV     CX,32               ;│
PT1LP:  MOVSW                       ;│ 指定位置にピーヨのデータをセット
        MOVSW                       ;│
        ADD     DI,80-4             ;│
        LOOP    PT1LP               ;│
        ROL     AX,1                ;│
        RET                         ;┘
PUT1D   ENDP

PUTP2   PROC
        MOV     SI,OFFSET BKDAT     ;┐
        MOV     BX,PATAD            ;│
        ADD     BX,4*32             ;│
        MOV     AX,BLUE             ;│
        CALL    P2ALL               ;│
        MOV     AX,RED              ;│ 背景で消去しながら新たなピーヨを
        CALL    P2ALL               ;│ 表示
        MOV     AX,GREEN            ;│
        CALL    P2ALL               ;│
        MOV     AX,ITSTY            ;│
```

```
        CALL    P2ALL                       ;
        RET                                 ;
PUTP2   ENDP

P2ALL   PROC
        MOV     ES,AX                       ;
        MOV     DI,3C00H+26H                ;
        MOV     BP,PATAD                    ;
        MOV     CX,32                       ;
P2AL0:  LODSW                               ;
        AND     AX,CS:[BP]                  ;
        OR      AX,[BX]                     ;
        STOSW                               ;
        ADD     BP,2                        ;
        ADD     BX,2                        ; プレーン別に背景＋くり抜き＋表示を
        LODSW                               ; 行う
        AND     AX,CS:[BP]                  ;
        OR      AX,[BX]                     ;
        STOSW                               ;
        ADD     BP,2                        ;
        ADD     BX,2                        ;
        ADD     DI,80-4                     ;
        LOOP    P2AL0                       ;
        RET                                 ;
P2ALL   ENDP

PUTP3   PROC
        CALL    EGC_ON                      ;
        OUTEGC  4A2H,0FFH                   ;
        OUTEGC  4A4H,0CF0H                  ;
        OUTEGC  4A8H,0FFFFH                 ;
        OUTEGC  4ACH,0                      ;
        OUTEGC  4AEH,<8*4-1>                ; G.VRAMの余りに背景を作成
        MOV     SI,OFFSET BKDAT             ;
        MOV     AX,0FFFEH                   ;
        CALL    DI7D0                       ;
        CALL    DI7D0                       ;
        CALL    DI7D0                       ;
        CALL    DI7D0                       ;
        OUTEGC  4A4H,0CC0H                  ;
        MOV     SI,PATAD                    ;
        MOV     AX,0FFF0H                   ;
        CALL    DI7D0                       ;
        OUTEGC  4A4H,0CFCH                  ; G.VRAMの余りにピーヨを重ね合わせる
        MOV     AX,0FFFEH                   ;
        CALL    DI7D0                       ;
        CALL    DI7D0                       ;
        CALL    DI7D0                       ;
        CALL    DI7D0                       ;
        OUTEGC  4A0H,0FFF0H                 ;
        OUTEGC  4A4H,28F0H                  ;
        MOV     SI,7D00H                    ;
        MOV     DI,3C00H+2EH                ; G.VRAMの余りから指定位置にピーヨを
        CALL    DI7D2                       ; 重ね合わせる
        CALL    EGC_OF                      ;
        RET                                 ;
PUTP3   ENDP

DI7D2   PROC
        PUSH    DS                          ;
        MOV     AX,BLUE                     ;
        MOV     DS,AX                       ;
        MOV     CX,32                       ;
```

```
PT3LP:  MOVSW                                   ;⎫
        MOVSW                                   ;⎬ EGCを利用してデータを転送する
        ADD     DI,80-4                         ;⎪
        LOOP    PT3LP                           ;⎪
        POP     DS                              ;⎪
        RET                                     ;⎭
DI7D2   ENDP

DI7D0   PROC
        MOV     DX,4A0H                         ;⎫
        OUT     DX,AX                           ;⎪
        MOV     DI,7D00H                        ;⎪
        MOV     CX,2*32                         ;⎬ 7D00Hへ128バイト転送
        REP     MOVSW                           ;⎪
        ROL     AX,1                            ;⎪
        RET                                     ;⎭
DI7D0   ENDP

TXCLR   LABEL BYTE                              ;テキスト・クリア ＆ 文字の属性の保存
        DB      1BH,"[2J"
        DB      1BH,"[s",1BH,"[>5h",1BH,"[1>h$"

TKEEP   LABEL BYTE                              ;文字の属性の復元
        DB      1BH,"[u",1BH,"[>51",1BH,"[1>1$"

BKDAT   LABEL BYTE                              ;背景データ・アドレス
        DB      4*32*4 DUP(?)

RVDAT   LABEL BYTE                              ;左右反転データアドレス
        DW      280H DUP(?)

CODE    ENDS

STACK   SEGMENT STACK                           ;スタック・セグメントの定義
        DW      100H    DUP(0)
STACK   ENDS

BLUE    SEGMENT AT 0A800H                       ;B面のセグメントの定義
BLUE    ENDS

RED     SEGMENT AT 0B000H                       ;R面のセグメントの定義
RED     ENDS

GREEN   SEGMENT AT 0B800H                       ;G面のセグメントの定義
GREEN   ENDS

ITSTY   SEGMENT AT 0E000H                       ;I 面のセグメントの定義
ITSTY   ENDS

        END
```

ピーヨの表示方法は、画面左側から順に次のようになっていますが、おそらくその違いを画面から直接判別することはできないでしょう。

① 背景消去→ピーヨ表示（EGC使用）：面アクセス回数＝9回
② 背景＋ピーヨ同時表示（ノーマルモード）：面アクセス回数＝4回
③ グラフィックVRAMのオフセットアドレス 7D00H以降へ、①と同じ方法（アドレスは連続）でデータ作成→表示位置へデータを4面同時転送（EGC使用）：面アクセス回数＝10回

このプログラムによる実験では、消去用背景とキャラクタのサイズが同一かつ小さいので、目に見える違いはありません。しかし、種類があれば優劣がつくのは世の常です。一見すると②が最も効率が良さそうですが、データ作成に要する時間が想像以上にかかっています。そこで、最終的にグラフィックVRAM 4 面の横 1 列（4×4 バイト）を完成させるのに要するクロック数を、プログラムから計算し比較してみましょう。

なお、グラフィックVRAMの次ラインへの加算時間（アドレス計算＋カウンタ処理）は含みますが、セグメント切り換えやモード切り換えに要する時間などその他の細かい部分は無視しています。

	①	②	③
1ライン処理 次ライン計算	36×9 21×9	130×4 21×4	9＋34×9＋36 21×1
トータル	513	604	372

一番効率の悪そうな③の方法が、意外や一番速度効率がいいという結果です。このことは、理想的なアルゴリズムが必ずしもプログラムとしてはベストではないことを裏付けています。本来、アルゴリズムもプログラムも "Simple is best" なのですが、両立することが難しい場合はプログラムをシンプルにしたほうがいいということです。

最後に、このプログラムに含まれている細かなテクニックについて述べておきます。まず、プログラムの先頭部分で左右反転したキャラクタ・データを作成していますが、左右反転というのはデータを左右反転するだけでなく、その並びをも入れ替えなければなりません（図 2-5-3）。

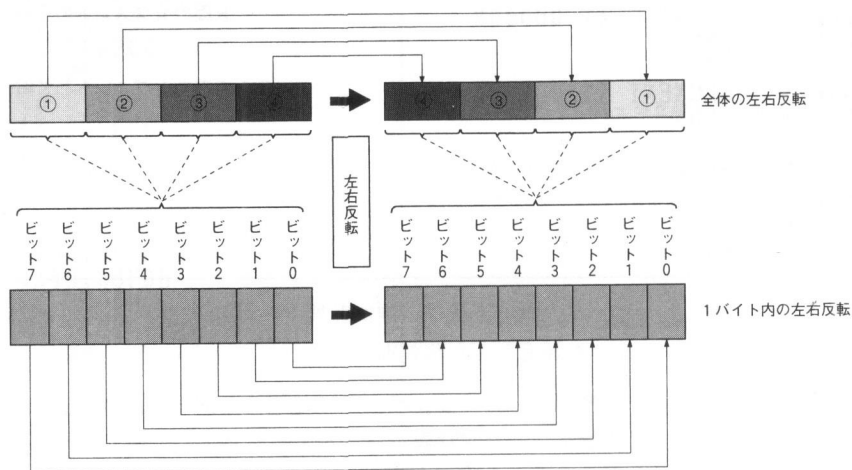



図 2-5-3　左右反転データの作成

データを左右反転する命令はありませんから、プログラムで左右反転データを作ることになります（ラベル REVRSの部分）。これはプログラムは簡単でも、1バイトのデータであれば8回もループしますから結構実行時間のかかる作業です（1バイトにつき正味 156クロック）。今回は最初に左右反転キャラクタ・データを作成しましたが、もしもデータを左右反転しながら表示するという場合、これでは速度的に無理があります。そこで、1バイトを迅速に左右反転するテクニックを紹介しましょう。

▼ 最初に「MKRVD」をコール

```
REVDA   DB      100H  DUP(0)

MKRVD   PROC
        MOV     SI,OFFSET REVDA
        XOR     AX,AX
        JMP     REVL2
REVL0:  MOV     CX,8
REVL1:  SHL     AL,1
        RCR     BYTE PTR [SI],1
        LOOP    REVL1
REVL2:  INC     SI
        INC     AL
        JNZ     REVL0
        RET
MKRVD   ENDP
```

▼ 利用時（11クロックでOK）

```
          :
        MOV     BX,OFFSET REVDA
          :
GETRV:  XLAT
          :
```

（注）いずれも DS=CS と仮定します。

まず、0～FFHの左右反転データをメモリ上に作成します。これが、たったの11クロックで左右反転データを得るためのポイントです。つまり、「左右反転データのある相対アドレス＝左右反転したいデータ」となりますから、使用時の例に示されているように簡単に左右反転データが取り出せるのです。

従来の左右反転（156 クロック）に比べて約1/16ですから、速度的にはデータを左右反転しながら表示することも十分に可能です。ただし、左右反転は横1列の並びも反転（表示アドレスの逆行）させなければならず、状況によっては速度的な負担が大きすぎることもあるかもしれません。このあたりの微妙な判断は、その時々で臨機応変に対応するしかないでしょう。

メモリ節約と速度追求……。こういったバランス感覚も、マシン語ゲームにおいては欠かすことができないテクニックなのです。

第3章　画面処理とアイデア

よく『アイデアは無限』といいますが、PC-9801 シリーズのゲームにも実に多くのアイデアが隠されています。プログラミングに少しでも興味があるならば、ゲームを表現している画面処理のアイデアも見逃すわけにはいきません。ところで……。

人間には持って生まれた天分というものがあります。これは、ときには才能ともいわれますが、要するに自己を表現するのに最も適した手段という意味です。それが若くして発見できれば天才となり、一生発見できなければ凡人となるのです。音楽家、漫画家、スポーツ選手……ジャンルを問わず自分の世界を表現できる人は輝いています。もちろん、それは天分に「努力」という磨きをかけた結果であることは言うまでもありません。

とはいえ、それもこれも自分の天分を発見してからのお話です。もっと簡単に自分の天分を見極めることができれば、苦労も挫折も少なくなるのですが……。

実は、PC-9801 シリーズという機種は「プログラミングによる自己表現」の天分発見機なのです。1枚のテキスト画面とカラーグラフィック画面……。たったこれだけで、どこまで他人と違った自分の世界を表現することができるか、これはもはやマニュアルにはないアイデアの世界です。

もし、PC-9801 シリーズでオリジナルな画面処理ができるようになれば、それは「あなたの天分が発見された」ということかもしれません。

3−1　テキスト画面マスク

日常、何気なく使っていたものが、ある日突然貴重なものであることがわかり、一躍脚光を浴びることがあります。例えば、水不足になって初めて水はその価値を認められたわけですが、そうなると「湯水のごとく使う」という言葉は死語同然です。一方、オイルショックで石油の価値が見直されましたが、それまで意識せずに使っていた「油断」という言葉が、それとは逆に本来の意味で急浮上したのです。とかく、モノの価値というのは流動的です。石油に代わる新エネルギー源が発見されれば、「油断」だっていつ死語になるかもしれません。まさに油断大敵なのです。

従来、テキスト画面というのはグラフィック画面を利用するゲームにとって、あまり重要視されませんでしたが、色々なテクニックが開発されると共に、テキスト画面の存在価値はにわか高まりました。

といっても、調子に乗ってグラフィック画面のように存在をアピールしたのではボロが出てしまいます。テキスト画面は自らの存在を隠すことで、主役であるグラフィック画面をより一層引き立てるのが役目なのです。自分を隠してグラフィック画面も隠す……。つまり、グラフィック画面の部分マスクがテキスト画面の存在価値なのです。

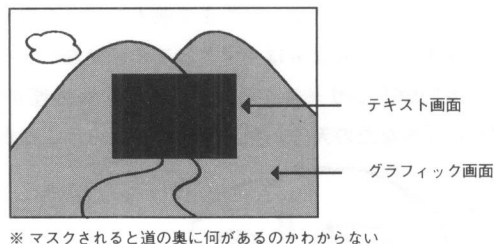



図 3-1-1　テキスト画面によるマスク

テキスト画面によるグラフィック画面マスクには、アイデア次第でいろいろな利用法が考えられます。基本的には見せたくないものにフタをするということですが、それだけにフタの存在を知られないようにするテクニックも要求されるわけです。ここでは、そういった隠すテクニックを隠さずに紹介していきます。

(1)　ブラインド効果

グラフィック画面の一部（または全部）を描き換えるとき、描く作業を見せたくない場合があります。そこで、テキスト画面でマスクをするわけですが、いきなり黒マスクをするのではなく、ブラインドが開閉するようにオシャレをしようというのです。

```
A>LIST3-1 ⏎
```

LIST 3-1

```
;************************************
;*            LIST3-1               *
;************************************

EXTRN   GSTAT:NEAR,GMOD8:NEAR,GDSET:NEAR,GCALL:NEAR

        PUBLIC  CCBOX

CODE    SEGMENT PUBLIC

        ASSUME  CS:CODE,DS:CODE

GETKEY  MACRO                               ;;1文字入力マクロ定義
        LOCAL   GLOOP
GLOOP:  MOV     DL,0FFH
        MOV     AH,6
        INT     21H
        JZ      GLOOP
        ENDM

PRINT   MACRO   STRING                      ;;文字列出力用マクロ定義
        MOV     DX,OFFSET STRING
        MOV     AH,09
        INT     21H
        ENDM

VRAMFUL MACRO   VSEG,DAT                    ;;指定プレーンをDATで埋める
        MOV     AX,VSEG
        MOV     ES,AX
        XOR     DI,DI
        MOV     AX,DAT
        MOV     CX,3E80H
        REP     STOSW
        ENDM

ESCKY   EQU     1BH                         ; ESC キーのアスキー・コード

PMAIN:  CLD
        CALL    GSTAT                       ;グラフィックスの開始
        JNB     $+5                         ;
        JMP     TEXIT                       ;異常終了であればTEXITへ
        CALL    GMOD8                       ;640×400,8色モード・セット
        JNB     $+5                         ;
        JMP     TEXIT                       ;異常終了であればTEXITへ
        PRINT   TXCLR                       ;テキスト画面クリア
        MOV     AL,1                        ;
        OUT     68H,AL                      ;簡易グラフとする
        VRAMFUL BLUE,0FFFFH                 ;┐
        VRAMFUL RED,0                       ;├ G.VRAMを初期データで埋める
        VRAMFUL GREEN,0                     ;┘
        MOV     SI,0                        ;
        CALL    CCBOX                       ;タイリング・ボックスを描く
DLOOP:  GETKEY                              ;1文字入力
        CMP     AL,"1"                      ;┐
        JNE     DLNT1                       ;│
        CALL    TCLS1                       ;│
        JMP     DLOOP                       ;│
```

```
DLNT1:  CMP     AL,"2"             ;} キー・チェック
        JNE     DLNT2              ;
        CALL    TCLS2              ;
        JMP     DLOOP              ;
DLNT2:  CMP     AL,"3"             ;
        JNE     DLNT3              ;
        CALL    TCLS3              ;
        JMP     DLOOP              ;
DLNT3:  CMP     AL,ESCKY           ;
        JE      KYESC              ;} [ESC] が押されていればKYESCへ
        JMP     DLOOP              ;
KYESC:  MOV     AX,CS              ;
        MOV     DS,AX              ;
        MOV     AX,0C00H           ;
        INT     21H                ;
        CALL    GCLS               ;} 画面をクリアしMS-DOSへ戻る
        PRINT   TKEEP              ;
        PRINT   TCLR2              ;
TEXIT:  MOV     AX,4C00H           ;
        INT     21H                ;

TCLS1   PROC
        MOV     BX,OFFSET CLDT1    ;
        MOV     AL,00000001B       ;} CLDT1で開閉中にタイリング・ボックスの
        MOV     ATRDT,AL           ;  カラーを変更する
        CALL    TVCS9              ;
        RET                        ;
TCLS1   ENDP

TCLS2   PROC
        MOV     BX,OFFSET CLDT2    ;
        MOV     AL,00000001B       ;} CLDT2で開閉中にタイリング・ボックスの
TVCC9:  MOV     ATRDT,AL           ;  カラーを変更する
        CALL    TVCS9              ;
        RET                        ;
TCLS2   ENDP

TCLS3   PROC
        MOV     BX,OFFSET CLDT3    ;
        MOV     AL,00010001B       ;
        MOV     ATRDT,AL           ;
        MOV     CH,8               ;
TC3L0:  CALL    TVCLS              ;
        INC     BX                 ;
        DEC     CH                 ;
        JNE     TC3L0              ;
        CALL    CCBOX              ;} CLDT3で開閉中にタイリング・ボックスの
        MOV     BX,OFFSET CLDT3    ;  カラーを変更する
        MOV     CH,8               ;
TC3L1:  MOV     AL,CS:[BX]         ;
        INC     BX                 ;
        NOT     AL                 ;
        CALL    TVACL              ;
        DEC     CH                 ;
        JNE     TC3L1              ;
        RET                        ;
TCLS3   ENDP

GCLS    PROC
        PUSH    DS                 ;
        CALL    GDSET              ;
        XOR     AX,AX              ;
        MOV     [SI],AX            ;
```

```
        MOV     [SI+2],AX           ;} グラフィック画面クリア
        MOV     SI,14               ;
        CALL    GCALL               ;
        POP     DS                  ;
        RET                         ;
GCLS    ENDP

TVCS9   PROC
        MOV     CX,8                ;
TVXL0:  CALL    TVCLS               ;} タイリング・ボックス部分を徐々に閉じる
        INC     BX                  ;
        LOOP    TVXL0               ;
        PUSH    BX                  ;
        CALL    CCBOX               ;} タイリング・ボックスのカラーを変更する
        POP     BX                  ;
        MOV     CX,8                ;
TVXL1:  DEC     BX                  ;
        CALL    TVCLS               ;
        LOOP    TVXL1               ;} タイリング・ボックス部分を徐々に開ける
        XOR     AL,AL               ;
        CALL    TVACL               ;
        RET                         ;
TVCS9   ENDP

TVCLS   PROC
        MOV     AL,CS:[BX]          ;
        CALL    TVACL               ;} タイリング・ボックスを指定データで埋める
        RET                         ;
TVCLS   ENDP

ATRDT   DB      0

TVACL   PROC
        PUSH    CX                  ;
        MOV     DI,4+160            ;
        XCHG    AX,DX               ;
        MOV     AX,TEXT             ;
        MOV     ES,AX               ;
        XCHG    AX,DX               ;
        MOV     AH,ATRDT            ;
        MOV     DX,20               ;
TVCL0:  MOV     CX,50               ;} タイリング・ボックス部分を指定データで
TVCL1:  MOV     ES:[DI+2000H],AH    ;  埋め、ウェイトを取る
        STOSB                       ;
        INC     DI                  ;
        LOOP    TVCL1               ;
        ADD     DI,160-100          ;
        DEC     DX                  ;
        JNE     TVCL0               ;
        MOV     CH,2                ;
        CALL    VWAIT               ;
        POP     CX                  ;
        RET                         ;
TVACL   ENDP

CCBOX   PROC
        MOV     AX,BLUE             ;
        MOV     ES,AX               ;
        MOV     AL,[SI+0]           ;
        CALL    CSBOX               ;
        MOV     AX,RED              ;
        MOV     ES,AX               ;
        MOV     AL,[SI+1]           ;
```

```
        CALL    CSBOX                   ;} タイリング・ボックスのカラーを変更
        MOV     AX,GREEN                ;  (SIの中身を乱数とする)
        MOV     ES,AX                   ;
        MOV     AL,[SI+2]               ;
        CALL    CSBOX                   ;
        INC     SI                      ;
        MOV     CH,8                    ;
        CALL    VWAIT                   ;
        RET                             ;
CCBOX   ENDP

VWAIT   PROC
        IN      AL,0A0H                 ;
        TEST    AL,00100000B            ;
        JNE     VWAIT                   ;
VWAT2:  IN      AL,0A0H                 ;
        TEST    AL,00100000B            ;} ウェイト用
        JE      VWAT2                   ;
        DEC     CH                      ;
        JNE     VWAIT                   ;
        RET                             ;
VWAIT   ENDP

CSBOX   PROC
        MOV     DI,80*16+2              ;
        MOV     DX,20*16                ;
CSBL0:  MOV     CX,50                   ;
CSBL1:  STOSB                           ;
        ROR     AL,1                    ;
        LOOP    CSBL1                   ;
        AND     BYTE PTR ES:[DI-1],0FEH ;} プレーン別にタイリング・ボックスのカラ
        ROR     AL,1                    ;  ーを変更
        ROR     AL,1                    ;
        ROR     AL,1                    ;
        ADD     DI,80-50                ;
        DEC     DX                      ;
        JNE     CSBL0                   ;
        RET                             ;
CSBOX   ENDP

CLDT1   LABEL BYTE                      ;テキスト開閉データ1
        DB      80H,81H,82H,83H,84H,85H,86H,87H

CLDT2   LABEL BYTE                      ;テキスト開閉データ2
        DB      88H,89H,8AH,8BH,8CH,8DH,8EH,87H

CLDT3   LABEL BYTE                      ;テキスト開閉データ3
        DB      1,11B,111B,1111B,10001111B
        DB      11001111B,11101111B,0FFH

TXCLR   LABEL BYTE                      ;テキスト・クリア & 文字の属性の保存
        DB      1BH,"[2J"
        DB      1BH,"[s",1BH,"[>5h",1BH,"[1>h$"

TKEEP   LABEL BYTE                      ;文字の属性の復元
        DB      1BH,"[u",1BH,"[>5l",1BH,"[1>l$"

TCLR2   LABEL BYTE                      ;テキスト画面クリア
        DB      1BH,"[2J$"

CODE    ENDS

STACK   SEGMENT STACK                   ;スタック・セグメントの定義
```

```
        DW       100H     DUP(0)
STACK   ENDS

TEXT    SEGMENT AT 0A000H                     ;テキスト・コードセグメントの定義
TEXT    ENDS

ATEXT   SEGMENT AT 0A200H                     ;アトリビュート・セグメントの定義
ATEXT   ENDS

BLUE    SEGMENT AT 0A800H                     ;B面のセグメントの定義
BLUE    ENDS

RED     SEGMENT AT 0B000H                     ;R面のセグメントの定義
RED     ENDS

GREEN   SEGMENT AT 0B800H                     ;G面のセグメントの定義
GREEN   ENDS

        END
```

プログラム（LIST 3-1）を実行すると、青い画面の一部分に妙なタイリングで塗られたボックスが現れます。ここで1～3キーのどれかを押すと、ボックスの部分が異なったタイプのブラインドで開閉します。ここでは、閉じている間にボックスのタイリングを変化させていますが、実際のゲームではこの間に絵を描くわけです。ブラインドの形状を変えたり、開閉パターンを変化させる（放射線状にする等）ことで、また違った雰囲気を出すことができます。いろいろと試してみてください。

(2) アニメーション用マスク

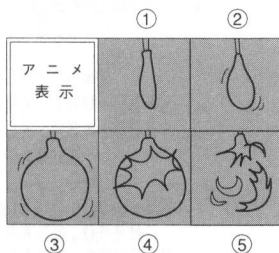
パソコンでアニメーションをするということは、結局グラフィック画面を描き換えるということです。当然、そのためにはデータが必要です。データはメインメモリに置くこともできますが、グラフィック・データは量が多いのでメモリの確保が大変です。そこで、グラフィックVRAMの一部に絵としてデータを置き、その部分は見えないようにテキスト画面でマスクをします。そして、拡張グラフィックス機能（4面同時転送）を使って転送すれば、アッという間に作画完了です。複数のパターンを用意すれば、簡単にアニメーションができます（図 3-1-2）。

・アニメ表示部以外は、パターンを描いてマスクする
・①～⑤の順でアニメ表示部へ転送すると、アニメーションとなる

図 3-1-2　テキスト画面と転送によるアニメーション

この方法（テキスト画面マスクと4面同時転送）は、アニメーションに限らず速度重視の画面処理には大変有効です。最近では、このテクニックゆえに成立しているゲームも少なくありません。もしゲーム画面がフル画面（640×400ドット）でなかっ

たり、あるいは画面のどこかに黒い部分があるという場合は、まず間違いなく見えない部分はデータだらけと判断していいでしょう。

次のプログラムは、このテクニックにその他の拡張グラフィックス機能をミックスした重ね合わせ処理です。2つのパターンが上下に移動し、その上をピーヨが左右に移動するという平凡な内容ですが、ぜひ非凡な応用法を考えてください。

```
A>LIST3-2 ↵
```

LIST 3-2

```
;***************************************
;*             LIST3-2                 *
;***************************************

EXTRN    GSTAT:NEAR,GMD16:NEAR,EGC_ON:NEAR,EGC_OF:NEAR
EXTRN    TMEN1:BYTE,GDSET:NEAR,GCALL:NEAR

CODE     SEGMENT PUBLIC

         ASSUME  CS:CODE,DS:CODE

OUTPORT MACRO    PORT,DATA                    ;;ポート出力用マクロ定義
        MOV      AX,DATA
        MOV      DX,PORT
        OUT      DX,AX
        ENDM

GETKEY  MACRO                                 ;;1文字入力マクロ定義
        LOCAL    GLOOP
GLOOP:  MOV      DL,0FFH
        MOV      AH,6
        INT      21H
        JZ       GLOOP
        ENDM

PRINT   MACRO    STRING                       ;;文字列出力用マクロ定義
        MOV      DX,OFFSET STRING
        MOV      AH,09
        INT      21H
        ENDM

MOV26M1 MACRO    MOD,LPC,SOU,DES              ;;データ転送用マクロ定義1
        MOV      AX,MOD
        MOV      BP,LPC
        MOV      SI,SOU
        MOV      DI,DES
        CALL     MOV_26
        ENDM

MOV26M2 MACRO    STR,DES2,LPC2                ;;データ転送用マクロ定義2
        LOCAL    MOVLP1
        MOV      DI,DES2
        MOV      DX,LPC2
MOVLP1: MOV      CX,13
        REP      STR
        ADD      DI,80-26
        DEC      DX
        JNE      MOVLP1
        ENDM
```

```
ESCKY   EQU     1BH                     ; ESC キーのアスキー・コード

PMAIN:  CLD
        CALL    GSTAT                   ;グラフィックスの開始
        JNB     $+5                     ;
        JMP     TEXIT                   ;異常終了であればTEXITへ
        CALL    GMD16                   ;640×400,16色モード・セット
        JNB     $+5                     ;
        JMP     TEXIT                   ;異常終了であればTEXITへ
        PRINT   TXCLR                   ;テキスト画面クリア
        CALL    GCLS                    ;グラフィック画面クリア
        PUSH    DS                      ;
        MOV     CX,15                   ;
TLP1:   PUSH    CX                      ;
        DEC     CX                      ;
        CALL    TRIANG                  ;
        POP     CX                      ; 三角形の描写
        ADD     WORD PTR YZAHYO1,9      ;
        ADD     WORD PTR XZAHYO2,7      ;
        SUB     WORD PTR YZAHYO2,4      ;
        SUB     WORD PTR XZAHYO3,7      ;
        SUB     WORD PTR YZAHYO3,4      ;
        LOOP    TLP1                    ;
        MOV     CX,16                   ;
TLP2:   PUSH    CX                      ;
        DEC     CX                      ;
        CALL    MARU                    ; 円形の描写
        POP     CX                      ;
        SUB     WORD PTR HANKEI,2       ;
        LOOP    TLP2                    ;
        PUSH    DS                      ;
        MOV     AX,BLUE                 ;
        CALL    TDTST                   ;
        MOV     AX,RED                  ;
        CALL    TDTST                   ; 重ね合わせ用データの作成
        MOV     AX,GREEN                ;
        CALL    TDTST                   ;
        MOV     AX,ITSTY                ;
        CALL    TDTST                   ;
        POP     DS                      ;
        MOV     AX,TEXT                 ;
        MOV     ES,AX                   ;
        XOR     DI,DI                   ;
        MOV     DX,0                    ;
        MOV     AL,87H                  ;
        MOV     CX,80*100H+25           ; テキスト画面マスク・セット
        CALL    TVBOX                   ;
        MOV     AL,0                    ;
        MOV     DI,26*2                 ;
        MOV     DX,160-28*2             ;
        MOV     CX,28*100H+25           ;
        CALL    TVBOX                   ;
        CALL    EGC_ON                  ;EGC拡張モード・オン
        OUTPORT 4A0H,0FFF0H             ;
        OUTPORT 4A2H,0FFH               ;
        OUTPORT 4A8H,0FFFFH             ;
        OUTPORT 4ACH,0                  ; EGC初期セット
        OUTPORT 4AEH,<8*26-1>           ;
        MOV     AX,BLUE                 ;
        MOV     ES,AX                   ;
TLOOP:  OUTPORT 4A4H,0CC0H              ;
        XOR     AX,AX                   ;
        MOV26M2 STOSW,54,400            ;
```

```
        MOV26M1 28F0H,200,0,MOAD1          ;} 重ね合わせデータ作成
        OUTPORT 4A4H,0C0CH                 ;
        MOV     SI,OFFSET KSANE            ;
        MOV26M2 MOVSW,MOAD2,200            ;
        MOV26M1 28FCH,200,80*200,MOAD2     ;
        OUTPORT 4A4H,0CC0H                 ;
        OUTPORT 4AEH,<8*4-1>               ;
        MOV     SI,OFFSET TMEN1            ;
        MOV     AX,0FFF0H                  ;
        CALL    DICHR                      ;
        MOV     AX,0CFCH                   ;
        MOV     DX,4A4H                    ;
        OUT     DX,AX                      ;
        MOV     AX,0FFFEH                  ;
        CALL    DICHR                      ;
        CALL    DICHR                      ;
        CALL    DICHR                      ;
        MOV     AX,PIYSW                   ;
        AND     AX,AX                      ;
        JNE     ST000                      ;} ピーヨの表示
        MOV     AX,PIYOF                   ;
        INC     AX                         ;
        CMP     AX,22*2                    ;
        JL      ST001                      ;
        MOV     WORD PTR PIYSW,1           ;
        JMP     ST001                      ;
ST000:  MOV     AX,PIYOF                   ;
        DEC     AX                         ;
        JNE     ST001                      ;
        MOV     WORD PTR PIYSW,0           ;
        XOR     AX,AX                      ;
ST001:  MOV     PIYOF,AX                   ;
        SHR     AX,1                       ;
        ADD     AX,PIYO0                   ;
        MOV     PIYOX,AX                   ;
        OUTPORT 4A0H,0FFF0H                ;
        OUTPORT 4AEH,<8*26-1>              ;} データ転送
        MOV26M1 28F0H,400,54,26            ;
        MOV     AX,ADNEX                   ;
        ADD     MOAD1,AX                   ;
        JNS     ST100                      ;
        SUB     MOAD1,AX                   ;
        NEG     ADNEX                      ;
        JMP     KSLP1                      ;
ST100:  SUB     MOAD2,AX                   ;
        JNS     KSLP1                      ;} 各図形の移動のためのデータを更新
        ADD     MOAD2,AX                   ;
        SUB     MOAD1,AX                   ;
        NEG     ADNEX                      ;
KSLP1:  MOV     AX,400H                    ;
        INT     18H                        ;
        AND     AH,1                       ;
        JNE     KYESC                      ;
        JMP     TLOOP                      ;
KYESC:  CALL    EGC_OF                     ;EGC拡張モード・オフ
        MOV     AX,CS                      ;
        MOV     DS,AX                      ;
        MOV     AX,0C00H                   ;
        INT     21H                        ;
        CALL    GCLS                       ;} 画面をクリアしてMS-DOSへ
        PRINT   TKEEP                      ;
        PRINT   TCLR2                      ;
TEXIT:  MOV     AX,4C00H                   ;
```

```
        INT     21H                     ;

GCLS    PROC
        PUSH    DS                      ;
        CALL    GDSET                   ;
        XOR     AX,AX                   ;
        MOV     [SI],AX                 ;
        MOV     [SI+2],AX               ; グラフィック画面クリア
        MOV     SI,14                   ;
        CALL    GCALL                   ;
        POP     DS                      ;
        RET                             ;
GCLS    ENDP

TVBOX   PROC
        PUSH    CX                      ;
TBOXL:  MOV     BYTE PTR ES:[DI+2000H],1;
        STOSB                           ;
        INC     DI                      ;
        DEC     CH                      ; テキスト画面に指定値でボックス作成
        JNE     TBOXL                   ;
        POP     CX                      ;
        ADD     DI,DX                   ;
        DEC     CL                      ;
        JNE     TVBOX                   ;
        RET                             ;
TVBOX   ENDP

TDTST   PROC
        MOV     DS,AX                   ;
        MOV     DI,OFFSET KSANE         ;
        MOV     SI,200*80               ;
        MOV     DX,200                  ;
LLP02:  MOV     CX,13                   ;
LLP03:  LODSW                           ;
        OR      CS:[DI],AX              ; 重ね合わせ用データの作成
        ADD     DI,2                    ;
        LOOP    LLP03                   ;
        ADD     SI,80-26                ;
        DEC     DX                      ;
        JNE     LLP02                   ;
        RET                             ;
TDTST   ENDP

MOV_26  PROC
        MOV     DX,4A4H                 ;
        OUT     DX,AX                   ;
        PUSH    DS                      ;
        MOV     AX,BLUE                 ;
        MOV     DS,AX                   ;
MOVLP0: MOV     CX,13                   ;
        REP     MOVSW                   ; データ転送
        ADD     SI,80-26                ;
        ADD     DI,80-26                ;
        DEC     BP                      ;
        JNE     MOVLP0                  ;
        POP     DS                      ;
        RET                             ;
MOV_26  ENDP

DICHR   PROC
        MOV     DX,04A0H                ;
        OUT     DX,AX                   ;
```

```
        MOV     DI,CS:PIYOX                    ;
        MOV     CX,32                          ;
DICLP:  MOVSW                                  ;
        MOVSW                                  ;
        ADD     DI,80-4                        ;  ピーヨの表示
        LOOP    DICLP                          ;
        ROL     AX,1                           ;
        RET                                    ;
DICHR   ENDP

TRIANG  PROC
        PUSH    DS                             ;
        PUSH    CX                             ;
        CALL    GDSET                          ;
        POP     CX                             ;
        MOV     AX,0                           ;
        MOV     [SI],AX                        ;
        MOV     [SI+2],AX                      ;
        MOV     [SI+4],AX                      ;
        MOV     AX,CS:XZAHYO1                  ;
        MOV     [SI+6],AX                      ;
        MOV     AX,CS:YZAHYO1                  ;
        MOV     [SI+8],AX                      ;
        MOV     AX,CS:XZAHYO2                  ;
        MOV     [SI+0AH],AX                    ;  三角形の描写
        MOV     AX,CS:YZAHYO2                  ;
        MOV     [SI+0CH],AX                    ;
        MOV     AX,CS:XZAHYO3                  ;
        MOV     [SI+0EH],AX                    ;
        MOV     AX,CS:YZAHYO3                  ;
        MOV     [SI+10H],AX                    ;
        MOV     AL,1                           ;
        MOV     [SI+20H],AL                    ;
        MOV     [SI+22H],CX                    ;
        MOV     SI,17                          ;
        CALL    GCALL                          ;
        POP     DS                             ;
        RET                                    ;
TRIANG  ENDP

MARU    PROC
        PUSH    DS                             ;
        PUSH    CX                             ;
        CALL    GDSET                          ;
        POP     CX                             ;
        MOV     AX,0                           ;
        MOV     [SI],AX                        ;
        MOV     [SI+2],AX                      ;
        MOV     [SI+4],AX                      ;
        MOV     AX,CS:MARUX1                   ;
        MOV     [SI+6],AX                      ;
        MOV     AX,CS:MARUY1                   ;  円形の描写
        MOV     [SI+8],AX                      ;
        MOV     AX,CS:HANKEI                   ;
        MOV     [SI+0AH],AX                    ;
        MOV     [SI+16H],CX                    ;
        MOV     AL,1                           ;
        MOV     [SI+22H],AL                    ;
        MOV     [SI+24H],CX                    ;
        MOV     SI,20                          ;
        CALL    GCALL                          ;
        POP     DS                             ;
        RET                                    ;
```

```
MARU     ENDP
XZAHY01  DW      100                           ;
YZAHY01  DW      8                             ;
XZAHY02  DW      1                             ;} 三角形の各頂点の座標
YZAHY02  DW      200                           ;
XZAHY03  DW      199                           ;
YZAHY03  DW      200                           ;

MARUX1   DW      100                           ;円の中心のX座標
MARUY1   DW      300                           ;円の中心のY座標
HANKEI   DW      99                            ;半径

PIYOX    DW      3C00H+54+10                   ;
PIYOO    DW      3C00H+54                      ;
PIYOF    DW      10*2                          ;} ピーヨ・データ・ワーク・エリア
PITCH    DW      1                             ;
PIYSW    DW      0                             ;
ADNEX    DW      80                            ;

MOAD1    DW      54                            ;} 図形移動用
MOAD2    DW      54+80*200                     ;

TXCLR    LABEL BYTE                            ;テキスト・クリア & 文字の属性の保存
         DB      1BH,"〔2J"
         DB      1BH,"〔s",1BH,"〔〉5h",1BH,"〔1〉h$"

TKEEP    LABEL BYTE                            ;文字の属性の復元
         DB      1BH,"〔u",1BH,"〔〉5l",1BH,"〔1〉l$"

TCLR2    LABEL BYTE                            ;テキスト・クリア
         DB      1BH,"〔2J$"

KSANE    LABEL WORD                            ;重ね合わせ用データ格納用
         DW      13*400 DUP(0)

CODE     ENDS

STACK    SEGMENT STACK                         ;スタック・セグメントの定義
         DW      100H    DUP(0)
STACK    ENDS

TEXT     SEGMENT AT 0A000H                     ;テキスト・コードセグメントの定義
TEXT     ENDS

BLUE     SEGMENT AT 0A800H                     ;B面のセグメントの定義
BLUE     ENDS

RED      SEGMENT AT 0B000H                     ;R面のセグメントの定義
RED      ENDS

GREEN    SEGMENT AT 0B800H                     ;G面のセグメントの定義
GREEN    ENDS

ITSTY    SEGMENT AT 0E000H                     ;I 面のセグメントの定義
ITSTY    ENDS

         END
```

メインのマスク処理は簡単ですが、拡張グラフィックスは選択されたモードと役割をよく理解することが大切です。特に、プログラムの先頭では、左下のパターンに対する抜き型データを作成していることに注意してください。このデータは重ね合わせ処理の時に元のVRAMのデータに対して、左下のパターンをくり抜くために使っています。

また、640×200ドットモードを選択すると、G.VRAMの半分が処理用に使え、しかもマスクの必要はありません。これを第2画面と組み合わせると、アイデア次第で用途は大きく広がります。ぜひトライしてみてください。

(3) マスク移動

テキスト画面のマスクだからといって、なにも固定して使わなければならないということはありません。テキスト画面マスクを動かしてグラフィック画面のほうを固定しても構わないし、あるいは両方動かしてもいいわけです。重要なのは、テキスト画面マスクの存在を隠すことではなく、いかにして画面を効果的に見せるかなのです。

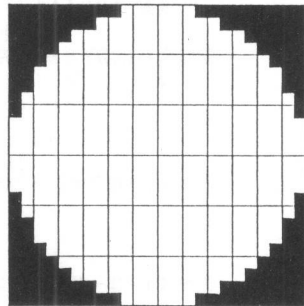
例えば、テキスト・マスクによるスポット処理なんていうのは、非常に簡単で効果的なものです。欠点は、テキスト・グラフィックは 160×100ドットと粗いので、円がゴツゴツしたものになってしまうことです（図 3-1-3）。

※ 1ドット（■）はグラフィック画面（640×400）で表すと4×4ドットに相当する。

図 3-1-3　テキスト・グラフィックでの円

しかし、その粗さが目立つのはグラフィック画面がベタで塗られているときで、黒が多ければほとんどわからなくなります。例えば、プログラム（LIST 3-3）を実行してみてください。

```
A>LIST3-3 ⏎
```

LIST 3-3

```
;***********************************
;*            LIST3-3              *
;***********************************

EXTRN   GSTAT:NEAR,GMOD8:NEAR,GDSET:NEAR,GCALL:NEAR

CODE    SEGMENT PUBLIC

        ASSUME  CS:CODE,DS:CODE

OUTPORT MACRO   PORT,DATA                   ;;ポート出力用マクロ定義
        MOV     AX,DATA
        MOV     DX,PORT
        OUT     DX,AX
        ENDM

GETKEY  MACRO                               ;;1文字入力用マクロ定義
        LOCAL   GLOOP
GLOOP:  MOV     DL,0FFH
        MOV     AH,6
        INT     21H
        JZ      GLOOP
        ENDM

PRINT   MACRO   STRING                      ;;文字列出力用マクロ定義
        MOV     DX,OFFSET STRING
        MOV     AH,09
        INT     21H
        ENDM

VRAMCLS MACRO   VSEG                        ;;VRAMクリア用マクロ定義
        MOV     AX,VSEG
        MOV     ES,AX
        MOV     CX,4000H
        XOR     AX,AX
        MOV     DI,AX
        REP     STOSW
        ENDM

ESCKY   EQU     1BH                         ; [ESC] キーのアスキー・コード

PMAIN:  CLD                                 ;
        CALL    GSTAT                       ;グラフィックスの開始
        JNB     $+5                         ;
        JMP     TEXIT                       ;異常終了であればTEXITへ
        CALL    GMOD8                       ;640×400,8色モード・セット
        JNB     $+5                         ;
        JMP     TEXIT                       ;異常終了であればTEXITへ
        PRINT   TXCLR                       ;テキスト画面クリア
        VRAMCLS BLUE                        ;
        VRAMCLS RED                         ;} グラフィック画面クリア
        VRAMCLS GREEN                       ;
        MOV     AL,1                        ;
        OUT     68H,AL                      ;簡易グラフィック選択
        MOV     AX,TEXT                     ;
        MOV     ES,AX                       ;
        MOV     DI,2000H                    ;} アトリビュートの設定
        MOV     CX,80*25                    ;
        MOV     AX,11H                      ;
        REP     STOSW                       ;
        MOV     AX,0FFH                     ;
```

```
        MOV     DI,0                    ;} テキスト画面マスク
        MOV     CX,80*25                ;
        REP     STOSW                   ;
        CALL    DSPLN                   ;放射状に線を引く
        MOV     CX,WORD PTR TXPOS       ;
        CALL    PUTEN                   ;指定位置にスポットを表示する
MLOOP:  GETKEY                          ;
        MOV     CX,WORD PTR TXPOS       ;
        CMP     AL,"2"                  ;
        JNE     MLP01                   ;
        INC     CL                      ;
        CMP     CL,39                   ;
        JNE     CPTEN                   ;
        JMP     MLOOP                   ;
MLP01:  CMP     AL,"4"                  ;
        JNE     MLP02                   ;
        DEC     CH                      ;
        JNS     CPTEN                   ;
        JMP     MLOOP                   ;
MLP02:  CMP     AL,"6"                  ;} ②④⑥⑧により、エンドチェックをしながら
        JNE     MLP03                   ;  スポットの座標を動かす
        INC     CH                      ;
        CMP     CH,69                   ;
        JNE     CPTEN                   ;
        JMP     MLOOP                   ;
MLP03:  CMP     AL,"8"                  ;
        JNE     MLP04                   ;
        DEC     CL                      ;
        JNS     CPTEN                   ;
        JMP     MLOOP                   ;
MLP04:  CMP     AL,ESCKY                ;
        JE      TOBSC                   ;
        JMP     MLOOP                   ;
CPTEN:  PUSH    CX                      ;
        CALL    VWAIT                   ;
        MOV     CX,WORD PTR TXPOS       ;
        CALL    CLSEN                   ;} 旧スポットをマスクし、新しいスポットを表示
        POP     CX                      ;  する
        MOV     WORD PTR TXPOS,CX       ;
        CALL    PUTEN                   ;
        JMP     MLOOP                   ;
TOBSC:  PUSH    ES                      ;
        VRAMCLS BLUE                    ;} グラフィックス画面クリア
        VRAMCLS RED                     ;
        VRAMCLS GREEN                   ;
        POP     ES                      ;
        XOR     DI,DI                   ;
        MOV     CX,80*25                ;
        MOV     AX,0                    ;
        REP     STOSW                   ;} アトリビュートを元に戻す
        MOV     DI,2000H                ;
        MOV     CX,80*25                ;
        MOV     AX,0E1H                 ;
        REP     STOSW                   ;
        PRINT   TKEEP                   ;テキストの属性の復元
TEXIT:  MOV     AX,4C00H                ;リターン・コード・セット
        INT     21H                     ;MS-DOSへ

TXPOS   DB      0,0                     ;スポットの座標（X,Y）

CLSEN   PROC
        CALL    TXADR                   ;
        MOV     AX,0FFH                 ;
```

```
        MOV     DL,CL                           ;
        MOV     DH,0                            ;
CENL0:  MOV     CX,12                           ;
        REP     STOSW                           ;
        ADD     DI,160-12*2                     ;  スポットをマスクする
        DEC     DX                              ;
        JNE     CENL0                           ;
        RET                                     ;
CLSEN   ENDP

PUTEN   PROC
        CALL    TXADR                           ;CXからT.VRAM、データアドレス、縦列数を求める
        XOR     AX,AX                           ;
        XOR     DX,DX                           ;
        MOV     DL,CL                           ;
PENL0:  MOV     CX,6                            ;  スポット（一列の左側）表示
PENL1:  LODSB                                   ;
        STOSW                                   ;
        LOOP    PENL1                           ;
        PUSH    SI                              ;
        MOV     CX,6                            ;
PENL2:  DEC     SI                              ;
        MOV     AL,[SI]                         ;
        ROL     AL,1                            ;
        ROL     AL,1                            ;  スポット（一列の右側）表示
        ROL     AL,1                            ;
        ROL     AL,1                            ;
        STOSW                                   ;
        LOOP    PENL2                           ;
        POP     SI                              ;
        ADD     DI,160-12*2                     ;
        DEC     DX                              ;  縦列分を表示してスポットを完成
        JNE     PENL0                           ;
        RET                                     ;
PUTEN   ENDP

TXADR   PROC                                    ;
        XOR     BX,BX                           ;
        MOV     BL,CL                           ;
        SHR     BX,1                            ;
        MOV     AX,160                          ;
        MUL     BX                              ;
        XCHG    AX,DI                           ;
        MOV     AL,CH                           ;
        CBW                                     ;  CXからT.VRAM（DI）、データアドレス（SI）
        SHL     AX,1                            ;  縦列数CLを求める
        ADD     DI,AX                           ;
        MOV     SI,OFFSET CDAT1                 ;
        SHR     CL,1                            ;
        MOV     CL,6                            ;
        JNB     TXART                           ;
        MOV     SI,OFFSET CDAT2                 ;
        INC     CL                              ;
TXART:  RET
TXADR   ENDP

VWAIT   PROC
        IN      AL,0A0H                         ;
        TEST    AL,00100000B                    ;
        JNE     VWAIT                           ;
VWAT2:  IN      AL,0A0H                         ;  ウェイト用
        TEST    AL,00100000B                    ;
        JE      VWAT2                           ;
```

```
        RET                                 ;」
VWAIT   ENDP

DSPLN   PROC                                ;┐
        MOV     CX,0                        ;│
DSPL1:  MOV     LINEX1,CX                   ;│
        MOV     WORD PTR LINEY1,0           ;│
        MOV     AX,638                      ;│
        SUB     AX,CX                       ;│
        MOV     LINEX2,AX                   ;│
        MOV     WORD PTR LINEY2,399         ;│
        MOV     AX,CX                       ;│
        AND     AX,7                        ;│
        MOV     COLOR,AX                    ;│
        CALL    LINE                        ;│
        ADD     CX,19                       ;│
        CMP     CX,638                      ;├ 画面に放射状に線を引く
        JLE     DSPL1                       ;│
        MOV     CX,0                        ;│
DSPL2:  MOV     WORD PTR LINEX1,0           ;│
        MOV     LINEY1,CX                   ;│
        MOV     WORD PTR LINEX2,638         ;│
        MOV     AX,399                      ;│
        SUB     AX,CX                       ;│
        MOV     LINEY2,AX                   ;│
        MOV     AX,CX                       ;│
        AND     AX,7                        ;│
        MOV     COLOR,AX                    ;│
        CALL    LINE                        ;│
        ADD     CX,18                       ;│
        CMP     CX,399                      ;│
        JLE     DSPL2                       ;│
        RET                                 ;┘
DSPLN   ENDP

LINE    PROC                                ;┐
        PUSH    BX                          ;│
        PUSH    CX                          ;│
        PUSH    DS                          ;│
        CALL    GDSET                       ;│
        MOV     AX,0                        ;│
        MOV     [SI],AX                     ;│
        MOV     [SI+2],AX                   ;│
        MOV     [SI+4],AX                   ;│
        MOV     AX,CS:LINEX1                ;│
        MOV     [SI+6],AX                   ;│
        MOV     AX,CS:LINEY1                ;│
        MOV     [SI+8],AX                   ;│
        MOV     AX,CS:LINEX2                ;├ 直線の描写
        MOV     [SI+0AH],AX                 ;│
        MOV     AX,CS:LINEY2                ;│
        MOV     [SI+0CH],AX                 ;│
        MOV     AL,0                        ;│
        MOV     [SI+0EH],AL                 ;│
        MOV     AX,CS:COLOR                 ;│
        MOV     [SI+10H],AX                 ;│
        MOV     SI,16                       ;│
        CALL    GCALL                       ;│
        POP     DS                          ;│
        POP     CX                          ;│
        POP     BX                          ;│
        RET                                 ;┘
LINE    ENDP
```

```
LINEX1  DW      0                       ;直線の始点のX座標
LINEY1  DW      0                       ;直線の始点のY座標
LINEX2  DW      0                       ;直線の終点のX座標
LINEY2  DW      0                       ;直線の終点のY座標
COLOR   DW      0                       ;直線の色

CDAT1   LABEL BYTE                      ;（X座標）＝偶数時のスポット・データ
        DB      0FFH,0FFH,37H,13H,1,0
        DB      7FH,1,0,0,0,0
        DB      1,0,0,0,0,0
        DB      8,0,0,0,0,0
        DB      0EFH,8,0,0,0,0
        DB      0FFH,0FFH,0CEH,8CH,8,0

CDAT2   LABEL BYTE                      ;（X座標）＝奇数時のスポット・データ
        DB      0FFH,0FFH,0FFH,7FH,37H,33H
        DB      0FFH,37H,1,0,0,0
        DB      17H,0,0,0,0,0
        DB      0,0,0,0,0,0
        DB      8CH,0,0,0,0,0
        DB      0FFH,0CEH,8,0,0,0
        DB      0FFH,0FFH,0FFH,0EFH,0CEH,0CCH

TXCLR   LABEL BYTE                      ;テキスト・クリア & 文字の属性の保存
        DB      1BH,"[2J"
        DB      1BH,"[s",1BH,"[>5h",1BH,"[1>h$"

TKEEP   LABEL BYTE                      ;文字の属性の復元
        DB      1BH,"[u",1BH,"[>5l",1BH,"[1>l$"

CODE    ENDS

STACK   SEGMENT STACK                   ;スタック・セグメントの定義
        DW      100H    DUP(0)
STACK   ENDS

TEXT    SEGMENT AT 0A000H               ;テキスト・コードセグメントの定義
TEXT    ENDS

BLUE    SEGMENT AT 0A800H               ;B面のセグメントの定義
BLUE    ENDS

RED     SEGMENT AT 0B000H               ;R面のセグメントの定義
RED     ENDS

GREEN   SEGMENT AT 0B800H               ;G面のセグメントの定義
GREEN   ENDS

        END
```

移動時には多少ゴツゴツしますが、停止時にはこのスポットがテキスト画面によるマスクとはまず判断できません。どんなに高度なテクニックも、無条件に使用できるわけではないし、簡単なテクニックで済むものは簡単に済ませるべきです。大切なのは、状況を的確に把握し、それに合わせたテクニックを選択することなのです。

ところで、このプログラム（LIST 3-3）では、メモリ節約のためスポットの表示データは左半分用しか持っていません。右半分は、プログラム（ROL×4回）で作っ

ています。テキスト・グラフィックにおける左右反転はどうなっているか、図 1-2-3を参考にしながら確認してください。また、ここではスポットを４ドット単位で動かしていますから、Ｙ座標の偶数／奇数でデータを変えなければなりません。これは、テキスト・グラフィックならではの面倒な部分といえるでしょう。

　テキスト画面の利用法は、ここに紹介したものがすべてではありません。グラフィック画面を見えなくするだけでなく、表示している自分自身を見えなくしてもいいのです。もちろん、邪魔だから見えなくするのではなく、見せたくないから見えなくするということです。文章ではわかりにくいかもしれませんが、カラー０の文字とグラフィック画面を重ね合わせるとか、オフにしたテキスト画面の内容を参照する……とか、アイデア次第で利用法は無限です。”ないもの”を利用することはできませんが、せっかく”あるもの”を利用しない手はありません。テキスト画面を大いに活用しましょう。

3－2　プレーンの分割（その1）

アイデアというと、すぐ「アイデア→特許大発明→大金持ち→自分には無理」と連想してしまいますが、頭のいい人は他人のアイデアを応用して新しいアイデアを考えるそうです。聞くところによると、それが小発明のコツであるとか……。

88版の『マシン語ゲームプログラミング』を読んだ方であれば、『スカイ・ブルーザー』というタイトルのスクロール・シューティング・ゲームをご存じだと思います。あの旧式のゲームは、プレーンを分割して使うことによってスクロールと重ね合わせ処理を成立させていました（98版『マシン語ゲームプログラミング』の『スカイ・ブルーザー』はフルカラー仕様の別テクニック）。

時代に逆行するようですが、ここで改めて88版『スカイ・ブルーザー』の基本を振り返ってみることにしましょう。

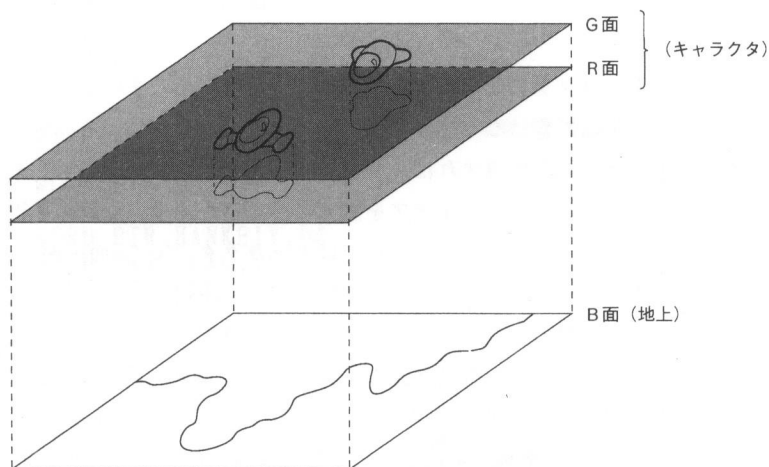



図 3-2-1　『スカイ・ブルーザー』の画面処理

図 3-2-1を見てください。スクロールする地上背景はB面で、自機および襲来する謎の飛行物体はR／G面を使って表示しています。こうすれば、地上は地上で勝手にスクロールさせることができるし、キャラクタは地上との重ね合わせ処理を気にせずに表示することができます。もちろん、単に使用するプレーンを分けただけでは分割したことにはなりません。このままでは、キャラクタ（R／G面）が地上（B面）の影響を受け色変化を起こしてしまうからです。そこで、R／G面がB面から独立して発色されるよう、パレットを変更します（図 3-2-2）。

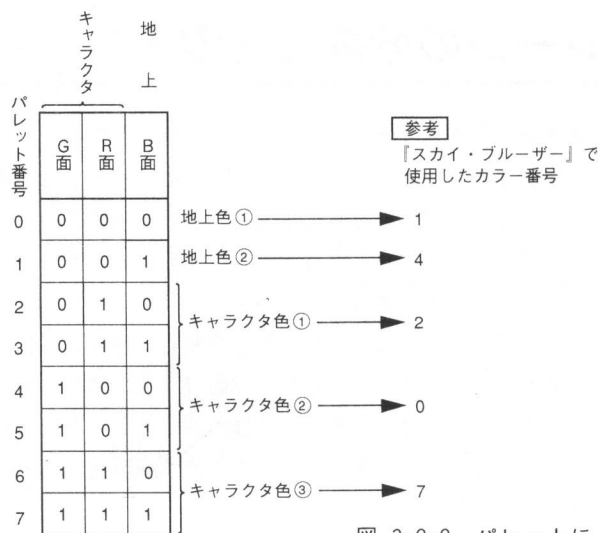



図 3-2-2 パレットによるプレーン分割

キャラクタに使用できる色数は3色になってしまいましたが、これで地上パターンによる影響はなくなります。参考として示されている88版『スカイ・ブルーザー』用のカラーを変更すれば、ゲームのイメージも変わるかもしれません。しかし、ここにちょっとしたアイデアを加えるとまったく違った効果を得ることができます。

例えば、このプレーン分割ではキャラクタ（R／G面）が地上（B面）より優先して表示されていますが、これは単にゲーム設定によるものです。実際にはB／R／G各面は同格の存在ですから、プレーンの優先順位はパレット変更次第なのです。とはいえ、パレット番号とB／R／Gのビット関係（図 3-2-2参照）から、G→R→Bと表示優先順位が低くなるように設定したほうがわかりやすいのは事実です。また、I面が使える機種であればさらに複雑な組み合わせを取ることもできます。

とりあえず、ここではB／R面でキャラクタを表示し、G面で上空を流れる雲を表してみました。まずは、プログラムを実行して88版『スカイ・ブルーザー』の雰囲気だけでも味わってください。実行プログラムは、(LIST3-4)です。

```
A>LIST3-4 ⏎
```

LIST3-4

```
;************************************
;*              LIST3-4             *
;************************************

EXTRN   GTERM:NEAR,GSTAT:NEAR,GMOD8:NEAR,GDSET:NEAR,GCALL:NEAR

CODE    SEGMENT PUBLIC

        ASSUME  CS:CODE,DS:CODE
```

```
PRINT   MACRO   STRING                          ;;文字列出力用マクロ定義
        MOV     DX,OFFSET STRING
        MOV     AH,09
        INT     21H
        ENDM

VRAMCLS MACRO   VSEG                            ;;VRAMクリア用マクロ定義
        MOV     AX,VSEG
        MOV     ES,AX
        MOV     CX,4000H
        XOR     AX,AX
        MOV     DI,AX
        REP     STOSW
        ENDM

PALSET  MACRO   D1,D2,D3,D4                     ;;パレット操作用マクロ定義
        MOV     AL,D1
        OUT     0AEH,AL
        MOV     AL,D2
        OUT     0AAH,AL
        MOV     AL,D3
        OUT     0ACH,AL
        MOV     AL,D4
        OUT     0A8H,AL
        ENDM

MCDAEN  MACRO   D1,D2,D3,D4,D5,D6,D7            ;;楕円描写用マクロ定義
        MOV     WORD PTR DAENX1,D1
        MOV     WORD PTR DAENY1,D2
        MOV     WORD PTR HANKEIX,D3
        MOV     WORD PTR HANKEIY,D4
        MOV     BYTE PTR FLAG,D5
        MOV     WORD PTR COLOR,D6
        CALL    DAEN
        ENDM

PMAIN:  CALL    GSTAT                           ;グラフィックスの開始
        JNB     $+5                             ;
        JMP     TEXIT                           ;異常終了であればTEXITへ
        CALL    GMOD8                           ;640×400,8色モード・セット
        JNB     $+5                             ;
        JMP     TEXIT                           ;異常終了であればTEXITへ
        PRINT   TXCLR                           ;テキスト画面クリア
        VRAMCLS BLUE                            ;
        VRAMCLS RED                             ;  グラフィック画面をクリア
        VRAMCLS GREEN                           ;
        PALSET  57H,7H,27H,67H                  ;パレットの変更
        MCDAEN  195,60,130,60,1,4               ;
        MCDAEN  305,100,120,30,1,4              ;  雲の表示
        MCDAEN  195,60,130,60,0,0               ;
        MCDAEN  420,145*2,100,50,1,4            ;
        CALL    CHRDI                           ;キャラクタの表示
MLOOP:  PUSH    DS                              ;
        MOV     AX,GREEN                        ;
        MOV     ES,AX                           ;
        MOV     DS,AX                           ;
        STD                                     ;
        MOV     SI,07CFEH                       ;
        MOV     DI,07CFEH+160                   ;
        MOV     AX,400                          ;
        MOV     CX,40*400                       ;  G面を下にループスクロールさせる
        REP     MOVSW                           ;
        MOV     SI,07CFEH+160                   ;
```

```
        MOV     CX,80                   ;
        REP     MOVSW                   ;
        POP     DS                      ;
        CLD                             ;
        MOV     CH,2                    ;ウェイトの回数
WAITL:  MOV     AX,0C00H                ;
        INT     21H                     ;
        MOV     AX,400H                 ;  ESC が押されていればTOBSCへ
        INT     18H                     ;
        TEST    AH,1                    ;
        JNE     TOBSC                   ;
        MOV     AX,406H                 ;
        INT     18H                     ;  SPACE が押されていなければUDKUMへ
        TEST    AH,10H                  ;
        JE      UDKUM                   ;
        MOV     AL,UPDSW                ;
        OR      AL,AL                   ;
        JNE     VWAIT                   ;必要に応じてパレットを変更する
        INC     AL                      ;(5→0,6→2,7→6)
        MOV     UPDSW,AL                ;
        PALSET  57H,0,22H,66H           ;
        JMP     VWAIT                   ;
UDKUM:  MOV     AL,UPDSW                ;
        OR      AL,AL                   ;
        JE      VWAIT                   ;必要に応じてパレットを変更する
        XOR     AL,AL                   ;(5→7,6→7,7→7)
        MOV     UPDSW,AL                ;
        PALSET  57H,7,27H,67H           ;
VWAIT:  IN      AL,0A0H                 ;
        TEST    AL,00100000B            ;
        JNE     VWAIT                   ;
VWAT2:  IN      AL,0A0H                 ;ウェイトを取ってMLOOPへ
        TEST    AL,00100000B            ;
        JE      VWAT2                   ;
        DEC     CH                      ;
        JNE     WAITL                   ;
        JMP     MLOOP                   ;
TOBSC:  PALSET  04H,15H,26H,37H         ;パレットの状態を初期の状態へ戻す
        PRINT   TKEEP                   ;テキストの属性を復元する
TEXIT:  CALL    GTERM                   ;グラフィックスの終了
        MOV     AX,4C00H                ;リターンコード・セット
        INT     21H                     ;MS-DOSへ

UPDSW   DB      0                       ;表示選択用

CHRDI   PROC
        MOV     SI,OFFSET CHRDT         ;
        MOV     AX,BLUE                 ;
        CLD                             ;
        CALL    CHLDI                   ;指定位置にキャラクタを表示
        MOV     AX,RED                  ;
        CALL    CHLDI                   ;
        RET                             ;
CHRDI   ENDP

CHLDI   PROC
        MOV     ES,AX                   ;
        MOV     DI,01F00H*2 + 5         ;
        MOV     CX,24                   ;
CHLLP:  MOV     DX,3                    ;
CHLL1:  LODSW                           ;
        MOV     ES:[DI+80],AX           ;指定プレーンにキャラクタデータを転送
        STOSW                           ;
```

```
        DEC     DX                  ;
        JNE     CHLL1               ;
        ADD     DI,160-6            ;
        LOOP    CHLLP               ;
        RET                         ;
CHLDI   ENDP

DAEN    PROC
        PUSH    DS                  ;
        PUSH    CX                  ;
        CALL    GDSET               ;
        POP     CX                  ;
        MOV     AX,0                ;
        MOV     [SI],AX             ;
        MOV     [SI+2],AX           ;
        MOV     [SI+4],AX           ;
        MOV     AX,CS:DAENX1        ;
        MOV     [SI+6],AX           ;
        MOV     AX,CS:DAENY1        ;
        MOV     [SI+8],AX           ;
        MOV     AX,CS:HANKEIX       ; 楕円を指定位置に描写する
        MOV     [SI+0AH],AX         ;
        MOV     AX,CS:HANKEIY       ;
        MOV     [SI+0CH],AX         ;
        MOV     AX,CS:COLOR         ;
        MOV     WORD PTR [SI+18H],AX ;
        MOV     AL,CS:FLAG          ;
        MOV     [SI+24H],AL         ;
        MOV     AX,CS:COLOR         ;
        MOV     WORD PTR [SI+26H],AX ;
        MOV     SI,21               ;
        CALL    GCALL               ;
        POP     DS                  ;
        RET                         ;
DAEN    ENDP

DAENX1  DW      195                 ;楕円の中心のX座標
DAENY1  DW      60                  ;楕円の中心のY座標
HANKEIX DW      130                 ;楕円のX方向半径
HANKEIY DW      60                  ;楕円のY方向半径
FLAG    DB      1                   ;塗りつぶしフラグ
COLOR   DW      4                   ;楕円の描写時のカラー

CHRDT   LABEL BYTE                  ;キャラクタ・データ
        DB      0,0,0FH,0F0H,0,0,0,0
        DB      3FH,0FCH,0,0,0,0,0FFH,0FFH
        DB      0,0,0,3,0FDH,77H,0C0H,0
        DB      0,7,0FEH,0FDH,60H,0,0,0FH
        DB      0FDH,0FEH,0B0H,0,0,1FH,0F5H,5FH
        DB      58H,0,0,3FH,0FFH,0FFH,0ACH,0
        DB      0,3FH,0FFH,0FFH,5CH,0,0,7FH
        DB      0F8H,0FH,0AEH,0,0,7FH,0E3H,0A7H
        DB      0D6H,0,0,0FFH,0CEH,83H,0EBH,0
        DB      0,0FFH,09EH,0C1H,0D7H,0,0,0FFH
        DB      0BAH,89H,0EBH,0,0,0FFH,0BAH,89H
        DB      0D7H,0,31H,0FFH,0B5H,29H,0EBH,8CH
        DB      6BH,0DFH,0AAH,51H,0D1H,0D6H,6FH,0DFH
        DB      0D6H,0A3H,0D1H,0F6H,6FH,0EFH,0F5H,4FH
        DB      0A3H,0F6H,0FFH,0F7H,0FFH,0FEH,87H,0FFH
        DB      0FFH,0FDH,0FFH,0FEH,1FH,0FBH,7FH,0FFH
        DB      0FFH,0F4H,0FFH,0FEH,0,0,0FFH,0ABH
        DB      0,0,0,0,3FH,0FCH,0,0
        DB      0,0,0,0,0,0,0,0
```

```
        DB      7,0E0H,0,0,0,0,1FH,0F8H
        DB      0,0,0,0,78H,3EH,0,0
        DB      0,1,0FEH,7FH,80H,0,0,3
        DB      0FCH,0FFH,0C0H,0,0,7,0F0H,1FH
        DB      0E0H,0,0,0FH,0FFH,0FFH,0F0H,0
        DB      0,0FH,0F8H,1FH,0F0H,0,0,1FH
        DB      0E0H,17H,0F8H,0,0,1FH,0C3H,0ABH
        DB      0F8H,0,0,3FH,8EH,85H,0FCH,0
        DB      0,3FH,1EH,0C2H,0FCH,0,0,3FH
        DB      3AH,82H,0FCH,0,0,3FH,3AH,82H
        DB      0FCH,0,0,7FH,35H,2,0FEH,0
        DB      10H,0DFH,2AH,2,0FFH,8,13H,0DFH
        DB      094H,5,0FDH,0C8H,1FH,0EFH,0C5H,43H
        DB      0FBH,0F8H,3FH,0F7H,0F0H,0FH,0F7H,0FCH
        DB      3FH,0FDH,0FFH,0FFH,0DFH,0FCH,0,0
        DB      0FFH,0FFH,0,0,0,0,3FH,0FCH
        DB      0,0,0,0,0,0,0,0

TXCLR   LABEL BYTE                              ;テキスト・クリア ＆ 文字の属性の保存
        DB      1BH,"[2J"
        DB      1BH,"[s",1BH,"[>5h",1BH,"[1>h$"

TKEEP   LABEL BYTE                              ;文字の属性の復元
        DB      1BH,"[u",1BH,"[>5l",1BH,"[1>l$"

CODE    ENDS

STACK   SEGMENT STACK                           ;スタック・セグメントの定義
        DW      100H    DUP(0)
STACK   ENDS

BLUE    SEGMENT AT 0A800H                       ;B面のセグメントの定義
BLUE    ENDS

RED     SEGMENT AT 0B000H                       ;R面のセグメントの定義
RED     ENDS

GREEN   SEGMENT AT 0B800H                       ;G面のセグメントの定義
GREEN   ENDS

        END
```

パレットの状態は図 3-2-3のようになっていますが、ここでパレット番号0に注目してください。パレット番号0というのは、B／R／Gどの面も発色していない地色のことですから、存在としてはB／R／G各面の下に位置すると考えることができます。したがって、パレット番号0は単独で地色としてもいいし、最も表示優先順位の低いプレーンとコンビにして使うこともできます。ちなみに、88版『スカイ・ブルーザー』ではB面（地上）と組み合わせ、地上を2色としていました。今回はキャラクタ表示用のB／R面が表示優先順位が低いので、パレット番号0はイメージとしては空あるいは海面なのですが、キャラクタ・カラーの一部としても使用することができます。

図3-2-3 プレーン分割の変化

もし、パレット番号0をキャラクタに使用しないならば、パレットを5→0、6→2、7→6と変更するだけでキャラクタが雲の上を通過するようになります。ここではキャラクタにパレット番号0を使っているので、その部分だけ雲の影響が出ますが、参考としてスペースキーを押している間だけパレットを変化させています。雲の上を通過するときに受ける影響と、プレーンの表示優先順位の関係を実際に確認してください。

いまにも敵機が現れそうな画面ですが、流れてくるのは同じ形の雲ばかりです。これが本物のゲームなら、雲の形状を変えたり濃霧を起こしたり……と、新たな障害物として楽しむことができるのですが、とりあえずは88版『スカイ・ブルーザー』との違いを感じられれば目的達成です。というのも、プレーン分割の応用はまだ続編があるからです。

今度は、画面を構成している雲/キャラクタという概念を変えてみましょう。キャラクタというと、どうしてもイメージが小さくなりますが、サイズの限定はどこにもありません。そこで、B/R面（パレット番号0を含んだ4色）を使って、フル画面で1枚の絵を描いてしまおうというのです。

こうなると、G面の役割は必然的にマスクしか残されていません。しかし、そのためにプレーン1枚を使うわけですから、たとえ複雑なタイトルロゴであろうと、形状を問わずにマスクをかけることができます。もちろん、かけたマスクを移動させることも自由自在です。サンプルとして、双眼鏡越しに絵を覗いたプログラムを示しますが、大きな絵の代わりにあの『おばけのピーヨ』を大量に出現させます。何匹出現しているか、バードウォッチングならぬピーヨウォッチングを楽しんでください。もちろん、表示色の制限がピーヨの表示にどう影響を与えたか、データの構造と表示カラーの関係を考えることも忘れないでください。また、I面が使える機種では、B/R/Gの3つのプレーンをキャラクタ用に、I面をマスク用に使う

ということもできます。

　では、プログラム（LIST3-5）を実行してください。双眼鏡は、テンキーで上下左右に移動することができます。

```
A>LIST3-5 ⏎
```

LIST 3-5

```
;************************************
;*              LIST3-5             *
;************************************

EXTRN   GTERM:NEAR,GSTAT:NEAR,GMOD8:NEAR,GDSET:NEAR,GCALL:NEAR,TMEN1:BYTE

CODE    SEGMENT PUBLIC

        ASSUME  CS:CODE,DS:CODE

GETKEY  MACRO                                   ;;1文字入力マクロ定義
        LOCAL   GLOOP
GLOOP:  MOV     DL,0FFH
        MOV     AH,6
        INT     21H
        JZ      GLOOP
        ENDM

PRINT   MACRO   STRING                          ;;文字列出力マクロ定義
        MOV     DX,OFFSET STRING
        MOV     AH,09
        INT     21H
        ENDM

VRAMCLS MACRO   VSEG,DATA                       ;;指定G.VRAMをDATAで埋める
        MOV     AX,VSEG
        MOV     ES,AX
        MOV     CX,4000H
        MOV     AX,DATA
        XOR     DI,DI
        REP     STOSW
        ENDM

PALSET  MACRO   D1,D2,D3,D4                     ;;パレット操作用マクロ定義
        MOV     AL,D1
        OUT     0AEH,AL
        MOV     AL,D2
        OUT     0AAH,AL
        MOV     AL,D3
        OUT     0ACH,AL
        MOV     AL,D4
        OUT     0A8H,AL
        ENDM

PMAIN:  CALL    GSTAT                           ;グラフィックスの開始
        JNB     $+5                             ;
        JMP     TEXIT                           ;異常終了であればTEXITへ
        CALL    GMOD8                           ;640×400,8色モード・セット
        JNB     $+5                             ;
        JMP     TEXIT                           ;異常終了であればTEXITへ
        PRINT   TXCLR                           ;テキスト画面クリア
```

```
        VRAMCLS BLUE,0                  ;|
        VRAMCLS RED,0                   ;} G.VRAMを初期データで埋める
        VRAMCLS GREEN,0FFFFH            ;|
        PALSET  10H,00,70H,60H          ;パレットの初期セット
        MOV     MARUX1,58               ;|
        MOV     MARUY1,58               ;
        MOV     HANKEI,50               ;
        XOR     CX,CX                   ;
        CALL    MARU                    ;} 双眼鏡作成
        MOV     MARUX1,125              ;
        MOV     MARUY1,58               ;
        MOV     HANKEI,50               ;
        XOR     CX,CX                   ;
        CALL    MARU                    ;|
        PUSH    DS                      ;|
        MOV     AX,GREEN                ;
        MOV     DS,AX                   ;
        MOV     AX,CS                   ;
        MOV     ES,AX                   ;
        MOV     SI,0                    ;
        MOV     DI,OFFSET SSDAT         ;} 双眼鏡のデータ作成
        MOV     AX,120                  ;
MSSDL:  MOV     CX,23                   ;
        REP     MOVSB                   ;
        ADD     SI,80-23                ;
        DEC     AX                      ;
        JNE     MSSDL                   ;
        POP     DS                      ;|
        MOV     SI,OFFSET DIPAD         ;|
        MOV     CX,20                   ;
CHRDL:  PUSH    CX                      ;
        MOV     DI,[SI]                 ;
        CALL    DIPHL                   ;} 全ピーヨを表示する
        INC     SI                      ;
        INC     SI                      ;
        POP     CX                      ;|
        LOOP    CHRDL                   ;
MLOOP:  GETKEY                          ;1文字入力
        CMP     AL,'2'                  ;|
        JNZ     NOTK2                   ;
        MOV     CX,100H                 ;
        JMP     MLOP1                   ;
NOTK2:  CMP     AL,'4'                  ;
        JNZ     NOTK4                   ;
        MOV     CX,0FFH                 ;} [2][4][6][8]のどれかが押されていたら
        JMP     MLOP1                   ;  MOVESを実行する
NOTK4:  CMP     AL,'6'                  ;
        JNZ     NOTK6                   ;
        MOV     CX,1                    ;
        JMP     MLOP1                   ;
NOTK6:  CMP     AL,'8'                  ;
        JNZ     WAIT                    ;
        MOV     CX,0FF00H               ;|
MLOP1:  CALL    MOVES                   ;
WAIT:   MOV     CH,2                    ;ウェイト
        CMP     AL,1BH                  ;
        JE      TOBSC                   ; [ESC] が押されていればTOBSCへ
VWAIT:  IN      AL,0A0H                 ;|
        TEST    AL,00100000B            ;
        JNE     VWAIT                   ;
VWAT2:  IN      AL,0A0H                 ;} ウェイト後MLOOPへ
        TEST    AL,00100000B            ;
        JE      VWAT2                   ;|
```

```
        DEC     CH                      ;
        JNE     VWAIT                   ;
        JMP     MLOOP                   ;
TOBSC:  PALSET  04H,15H,26H,37H         ;パレットの状態を元に戻す
        PRINT   TKEEP                   ;テキストの属性を元に戻す
TEXIT:  CALL    GTERM                   ;グラフィックスの終了
        MOV     AX,4C00H                ;リターン・コード・セット
        INT     21H                     ;MS-DOSへ

DIPAD   LABEL WORD                      ;ピーヨ表示位置データ
        DW      01F0H,03F0H,0782H,0862H
        DW      1225H,1240H,1943H,1E09H
        DW      2290H,2622H,25A0H,2C70H
        DW      32B8H,4142H,5CC0H,6060H
        DW      6730H,69D0H,6B20H,70C5H

SPOSX   DB      0                       ;双眼鏡X座標
SPOSY   DB      0                       ;双眼鏡Y座標

MOVES   PROC
        MOV     AL,SPOSX                ;
        ADD     AL,CL                   ;
        CMP     AL,80-23+1              ;
        JNB     MOVRT                   ;
        MOV     SPOSX,AL                ;  双眼鏡の移動後の座標を求める
        MOV     AL,SPOSY                ;
        ADD     AL,CH                   ;
        CMP     AL,50-15+1              ;
        JNB     MOVRT                   ;
        MOV     SPOSY,AL                ;
DIPSS:  MOV     CX,WORD PTR SPOSX       ;
        INC     CH                      ;
        MOV     DI,-280H                ;
        MOV     DX,280H                 ;
DIPL0:  ADD     DI,DX                   ;
        DEC     CH                      ;
        JNE     DIPL0                   ;
        ADD     DI,CX                   ;
        MOV     SI,OFFSET SSDAT         ;  双眼鏡を指定位置に表示
        MOV     AX,GREEN                ;
        MOV     ES,AX                   ;
        MOV     AX,78H                  ;
DIPL1:  MOV     CX,17H                  ;
        REP     MOVSB                   ;
        ADD     DI,39H                  ;
        DEC     AX                      ;
        JNE     DIPL1                   ;
MOVRT:  RET                             ;
MOVES   ENDP

DIPHL   PROC
        PUSH    SI                      ;
        MOV     SI,OFFSET TMEN1         ;
        MOV     AX,BLUE                 ;
        MOV     ES,AX                   ;
        CALL    DINXOR                  ;
        CALL    DIXOR                   ;  ピーヨを表示
        MOV     AX,RED                  ;
        MOV     ES,AX                   ;
        CALL    DIXOR                   ;
        POP     SI                      ;
        RET                             ;
DIPHL   ENDP
```

```
DIXOR   PROC
        PUSH    DI                  ;
        MOV     CX,32               ;
DIXLO:  LODSW                       ;
        XOR     ES:[DI],AX          ;
        ADD     DI,2                ; B面/R面とXORを取りながら
        LODSW                       ; ピーヨを表示
        XOR     ES:[DI],AX          ;
        ADD     DI,80-2             ;
        LOOP    DIXLO               ;
        POP     DI                  ;
        RET                         ;
DIXOR   ENDP

DINXOR  PROC
        PUSH    DI                  ;
        MOV     CX,32               ;
DINXLO: LODSW                       ;
        NOT     AX                  ;
        XOR     ES:[DI],AX          ;
        ADD     DI,2                ; B面にTMEN1のデータを反転して
        LODSW                       ; XORを取る
        NOT     AX                  ;
        XOR     ES:[DI],AX          ;
        ADD     DI,80-2             ;
        LOOP    DINXLO              ;
        POP     DI                  ;
        RET                         ;
DINXOR  ENDP

MARU    PROC
        PUSH    DS                  ;
        PUSH    CX                  ;
        CALL    GDSET               ;
        POP     CX                  ;
        MOV     AX,0                ;
        MOV     [SI],AX             ;
        MOV     [SI+2],AX           ;
        MOV     [SI+4],AX           ;
        MOV     AX,CS:MARUX1        ;
        MOV     [SI+6],AX           ;
        MOV     AX,CS:MARUY1        ; 指定した中心の座標に指定した半径で円を
        MOV     [SI+8],AX           ; 描写する
        MOV     AX,CS:HANKEI        ;
        MOV     [SI+0AH],AX         ;
        MOV     [SI+16H],CX         ;
        MOV     AL,1                ;
        MOV     [SI+22H],AL         ;
        MOV     [SI+24H],CX         ;
        MOV     SI,20               ;
        CALL    GCALL               ;
        POP     DS                  ;
        RET                         ;
MARU    ENDP

MARUX1  DW      0                   ;円の中心のX座標
MARUY1  DW      0                   ;円の中心のY座標
HANKEI  DW      0                   ;円の半径

TXCLR   LABEL BYTE                  ;テキスト・クリア ＆ 文字の属性の保存
        DB      1BH,"[2J"
        DB      1BH,"[s",1BH,"[>5h",1BH,"[1>h$"
```

```
TKEEP   LABEL BYTE                          ;文字の属性の復元
        DB      1BH,"〔u",1BH,"〔〉51",1BH,"〔1〉1$"

SSDAT   LABEL BYTE                          ;双眼鏡のデータ・エリア
        DB      120*23 DUP(0)

CODE    ENDS

STACK   SEGMENT STACK                       ;スタック・セグメントの定義
        DW      100H    DUP(0)
STACK   ENDS

BLUE    SEGMENT AT 0A800H                   ;B面のセグメントの定義
BLUE    ENDS

RED     SEGMENT AT 0B000H                   ;R面のセグメントの定義
RED     ENDS

GREEN   SEGMENT AT 0B800H                   ;G面のセグメントの定義
GREEN   ENDS

        END
```

これまた『スカイ・ブルーザー』とは違った雰囲気ですが、すべて基本は同じアイデアです。これをどう応用しどう発展させるか……、あなたのアイデアがそのカギを握っているのです。

3－3　プレーンの分割（その2）

手品師が「このハンカチにはタネも仕掛けもありません……」と言ってハンカチを胸のポケットから取り出しても、それをそのまま信じる人はいないでしょう。きっと秘密があるハズと、かえってハンカチを注目することになります。しかし、それではますます手品師の思うツボです。なぜなら、観客の目がハンカチに集まれば、もう片方の手はタネも仕掛けも思い通りに扱えるからです。

この場合、ハンカチは観客に見せるための存在であり、本当の手品の主役は見えないところで頑張っているもう片方の手です。とはいえ、それがバレるようでは手品師ではありません。たった1枚のハンカチを右に左に持ちかえては、どちらが主役の手かわからなくなるようにしているのです。

ところで、似たようなことを実行したプログラム（LIST 3-2）を覚えていますか。移動するパターンの重ね合わせ処理を、テキスト画面マスクと拡張グラフィックスの4面同時転送機能で実現したプログラムです。見えている部分がハンカチなら、見えない右側の作業はまさに主役の手そのものです。……が、このように役割が明確に分離していたのでは手品にはなりません。そこで、この節ではハンカチと主役の座が交互に入れ替わるような、そんな手品師的テクニックを披露することにしましょう。

前節でプレーンの分割をやりましたが、今回もテーマは同じプレーンの分割です。ただし、内容も用途もまったく違います。どのように違うか、それは次のプログラム（LIST 3-6）を実行すれば一目瞭然です。

```
A>LIST3-6 ⏎
```

LIST 3-6

```
;************************************
;*             LIST3-6              *
;************************************

EXTRN   GTERM:NEAR,GSTAT:NEAR,GMOD8:NEAR,GDSET:NEAR,GCALL:NEAR

CODE    SEGMENT PUBLIC

        ASSUME  CS:CODE,DS:CODE

PRINT   MACRO   STRING                          ;;文字列出力用マクロ定義
        MOV     DX,OFFSET STRING
        MOV     AH,09
        INT     21H
        ENDM

VRAMCLS MACRO   VSEG                            ;;VRAMクリア用マクロ定義
        MOV     AX,VSEG
        MOV     ES,AX
        MOV     CX,4000H
        XOR     AX,AX
        MOV     DI,AX
```

```
        REP     STOSW
        ENDM

PALSET  MACRO   D1,D2,D3,D4                     ;;パレット操作用マクロ定義
        MOV     AL,D1
        OUT     0AEH,AL
        MOV     AL,D2
        OUT     0AAH,AL
        MOV     AL,D3
        OUT     0ACH,AL
        MOV     AL,D4
        OUT     0A8H,AL
        ENDM

MBOX    MACRO   D1,D2,D3,D4,D5                  ;;BOX表示用マクロ定義
        MOV     WORD PTR COLOR,D5
        MOV     AX,BX
        ADD     AX,D1
        MOV     LINEX1,AX
        MOV     AX,CX
        ADD     AX,D2
        ADD     AX,AX
        MOV     LINEY1,AX
        MOV     AX,BX
        ADD     AX,D3
        MOV     LINEX2,AX
        MOV     AX,CX
        ADD     AX,D4
        ADD     AX,AX
        MOV     LINEY2,AX
        CALL    BOX
        ENDM

MLINE   MACRO   D1,D2,D3,D4,D5,D6,D7,D8,D9;;直線描写用マクロ定義
        MOV     WORD PTR COLOR,D9
        MOV     AX,BX
        ADD     AX,D1
        SUB     AX,D2
        MOV     LINEX1,AX
        MOV     AX,CX
        ADD     AX,D3
        ADD     AX,D4
        ADD     AX,AX
        MOV     LINEY1,AX
        MOV     AX,BX
        ADD     AX,D5
        SUB     AX,D6
        MOV     LINEX2,AX
        MOV     AX,CX
        ADD     AX,D7
        ADD     AX,D8
        ADD     AX,AX
        MOV     LINEY2,AX
        CALL    LINE
        ENDM

MDAEN   MACRO   D1,D2,D3,D4,D5                  ;;楕円描写用マクロ定義
        MOV     AX,BX
        ADD     AX,D1
        MOV     DAENX1,AX
        MOV     AX,CX
        ADD     AX,D2
        ADD     AX,AX
```

```
        MOV     DAENY1,AX
        MOV     HANKEIX,D3
        MOV     HANKEIY,D4
        ADD     HANKEIY,D4
        MOV     AX,D5
        MOV     COLOR,AX
        CALL    DAEN
        ENDM

PMAIN:  CALL    GSTAT                   ;グラフィックスの開始
        JNB     $+5                     ;
        JMP     TEXIT                   ;異常終了であればTEXITへ
        CALL    GMOD8                   ;640×400,8色モード・セット
        JNB     $+5                     ;
        JMP     TEXIT                   ;異常終了であればTEXITへ
        PRINT   TXCLR                   ;テキスト画面クリア
        VRAMCLS BLUE                    ;
        VRAMCLS RED                     ; グラフィックス画面クリア
        VRAMCLS GREEN                   ;
        MOV     CX,8                    ;
        MOV     BX,0                    ;
LP001:  PUSH    CX                      ;
        MOV     CX,0                    ;
        CALL    MKDAT                   ;
        POP     CX                      ;
        ADD     BX,80                   ;
        LOOP    LP001                   ; パターンの作成
        MOV     CX,6                    ;
        MOV     BX,0                    ;
LP002:  PUSH    CX                      ;
        MOV     CX,40                   ;
        CALL    MKDAT                   ;
        POP     CX                      ;
        ADD     BX,80                   ;
        LOOP    LP002                   ;
        CALL    MKP3D                   ;パターンデータの作成
        PALSET  00H,10H,04H,04H         ;パレットの変更
        MOV     AX,BLUE                 ;
        CALL    FLCLS                   ; B面に背景を作成
        CALL    BACKD                   ;
        CALL    DP3GP                   ;G面にパターンを作成し表示 (パレット変更)
MLOOP:  CALL    KSCAN                   ;移動用キースキャン
        XOR     BYTE PTR CS:SIGNX,1     ;パターン選択サインの更新
        JNE     DDP3G                   ;パターン選択が0でなければDDP3Gへ
        CALL    DP3RP                   ;R面にパターンを作成し表示 (パレット変更)
        JMP     SHORT MLOP1             ;MLOP1へ
DDP3G:  CALL    DP3GP                   ;G面にパターンを作成し表示 (パレット変更)
MLOP1:  MOV     DL,0FFH                 ;
        MOV     AH,06H                  ;
        INT     21H                     ; [ESC] が押されていなければMLOOPへ
        CMP     AL,1BH                  ;
        JNE     MLOOP                   ;
        PALSET  04H,15H,26H,37H         ;パレットの状態を初期の状態へ戻す
        PRINT   TKEEP                   ;テキストの属性を復元する
TEXIT:  CALL    GTERM                   ;グラフィックスの終了
        MOV     AX,4C00H                ;リターンコード・セット
        INT     21H                     ;MS-DOSへ

SIGNX   DB      0                       ;パターン選択用ワーク
PAT3D   DB      0                       ;パターン番号
XPO3D   DB      0                       ;パターンのX座標
YPO3D   DB      0                       ;パターンのY座標
```

```
MKP3D   PROC
        MOV     SI,0                          ;⎫
        MOV     DI,OFFSET DAT3D               ;⎪
        MOV     CX,8                          ;⎪
        CALL    M3DSB                         ;⎬ パターンデータを作成
        MOV     SI,80*80                      ;⎪
        MOV     CX,6                          ;⎪
        CALL    M3DSB                         ;⎪
        RET                                   ;⎭
MKP3D   ENDP

M3DSB   PROC
        PUSH    DS                            ;⎫
        MOV     AX,BLUE                       ;⎪
        MOV     DS,AX                         ;⎪
        MOV     AX,CS                         ;⎪
        MOV     ES,AX                         ;⎪
M3DSBL: PUSH    CX                            ;⎪
        PUSH    SI                            ;⎪
        MOV     AX,80                         ;⎪
M3DL0:  MOV     CX,5                          ;⎬ 画面からパターンデータを作成
        REP     MOVSW                         ;⎪
        ADD     SI,80-10                      ;⎪
        DEC     AX                            ;⎪
        JNE     M3DL0                         ;⎪
        POP     SI                            ;⎪
        ADD     SI,10                         ;⎪
        POP     CX                            ;⎪
        LOOP    M3DSBL                        ;⎪
        POP     DS                            ;⎪
        RET                                   ;⎭
M3DSB   ENDP

BACKD   PROC
        MOV     CX,400                        ;⎫
        MOV     BX,1                          ;⎪
BAKL0:  PUSH    CX                            ;⎪
        MOV     CH,39                         ;⎪
BAKL1:  MOV     BYTE PTR ES:[BX],1            ;⎪
        INC     BX                            ;⎪
        INC     BX                            ;⎪
        DEC     CH                            ;⎪
        JNE     BAKL1                         ;⎪
        INC     BX                            ;⎪
        INC     BX                            ;⎪
        POP     CX                            ;⎪
        LOOP    BAKL0                         ;⎬ 背景の表示
        MOV     BX,80*16                      ;⎪
        MOV     DX,80*15                      ;⎪
        MOV     CL,24                         ;⎪
BAKL2:  MOV     CH,80                         ;⎪
BAKL3:  MOV     BYTE PTR ES:[BX],0FFH         ;⎪
        INC     BX                            ;⎪
        DEC     CH                            ;⎪
        JNE     BAKL3                         ;⎪
        ADD     BX,DX                         ;⎪
        DEC     CL                            ;⎪
        JNE     BAKL2                         ;⎪
        RET                                   ;⎭
BACKD   ENDP

FLCLS   PROC
        MOV     ES,AX                         ;⎫
```

```
        MOV     CX,80*200                       ;⎫ 画面クリア用
        XOR     AX,AX                           ;⎪
        MOV     DI,AX                           ;⎪
        REP     STOSW                           ;⎪
        RET                                     ;⎭
FLCLS   ENDP

DP3RP   PROC
        MOV     AX,RED                          ;⎫
        CALL    DIP3D                           ;⎪ R面にパターンを作成し表示（パレット変更）
        PALSET  00H,11H,44H,44H                 ;⎪
        RET                                     ;⎭
DP3RP   ENDP

DP3GP   PROC
        MOV     AX,GREEN                        ;⎫
        CALL    DIP3D                           ;⎪ G面にパターンを作成し表示（パレット変更）
        PALSET  04H,14H,04H,14H                 ;⎪
        RET                                     ;⎭
DP3GP   ENDP

DIP3D   PROC
        MOV     OUTPO,AX                        ;⎫
        CALL    FLCLS                           ;⎪
        MOV     AL,PAT3D                        ;⎪
        INC     AL                              ;⎪
        CMP     AL,27                           ;⎪
        JB      PATOK                           ;⎪
        XOR     AL,AL                           ;⎪
PATOK:  MOV     PAT3D,AL                        ;⎪
        CMP     AL,14                           ;⎪
        JB      PTIA1                           ;⎪
        SUB     AL,27                           ;⎪
        NOT     AL                              ;⎪
PTIA1:  INC     AL                              ;⎪
        MOV     BX,OFFSET DAT3D-10*80           ;⎪
        MOV     DX,10*80                        ;⎪
GPADL:  ADD     BX,DX                           ;⎪
        DEC     AL                              ;⎪
        JNE     GPADL                           ;⎪
        PUSH    BX                              ;⎪
        MOV     CX,WORD PTR XPO3D               ;⎬ 指定プレーンにパターンを表示
        INC     CH                              ;⎪
        MOV     BX,-80*4                        ;⎪
        MOV     DX,80*4                         ;⎪
BCADL:  ADD     BX,DX                           ;⎪
        DEC     CH                              ;⎪
        JNE     BCADL                           ;⎪
        ADD     BX,CX                           ;⎪
        POP     DX                              ;⎪
        MOV     AX,80                           ;⎪
        MOV     ES,OUTPO                        ;⎪
        XCHG    SI,DX                           ;⎪
        XCHG    DI,BX                           ;⎪
DP3L0:  MOV     CX,10                           ;⎪
        REP     MOVSB                           ;⎪
        ADD     DI,80-10                        ;⎪
        DEC     AX                              ;⎪
        JNE     DP3L0                           ;⎪
        XCHG    SI,DX                           ;⎪
        XCHG    DI,BX                           ;⎪
        CALL    VWAIT                           ;⎪
        RET                                     ;⎭
```

```
DIP3D   ENDP

VWAIT   PROC
        IN      AL,0A0H                        ;
        TEST    AL,00100000B                   ;
        JNE     VWAIT                          ;
VWAT2:  IN      AL,0A0H                        ;  ウェイト用
        TEST    AL,00100000B                   ;
        JE      VWAT2                          ;
        RET                                    ;
VWAIT   ENDP

OUTPO   DW      BLUE                           ;G.VRAMセグメント値格納用

KSCAN   PROC
        MOV     CX,0                           ;
        MOV     AX,409H                        ;
        INT     18H                            ;
        TEST    AH,8                           ;
        JE      KSCN1                          ;
        MOV     CL,1                           ;
KSCN1:  TEST    AH,1                           ;
        JE      KSCN2                          ;
        MOV     CH,1                           ;
KSCN2:  MOV     AX,408H                        ;
        INT     18H                            ;
        TEST    AH,8                           ;
        JE      KSCN3                          ;
        MOV     CL,-1                          ;  2468をキースキャンし、パターンの座標を
KSCN3:  TEST    AH,40H                         ;  更新する
        JE      KSCN4                          ;
        MOV     CH,-1                          ;
KSCN4:  MOV     AL,XPO3D                       ;
        ADD     AL,CH                          ;
        CMP     AL,71                          ;
        JNB     YMCHK                          ;
        MOV     XPO3D,AL                       ;
YMCHK:  MOV     AL,YPO3D                       ;
        ADD     AL,CL                          ;
        CMP     AL,81                          ;
        JNB     KSRET                          ;
        MOV     YPO3D,AL                       ;
KSRET:  RET                                    ;
KSCAN   ENDP

MKDAT   PROC
        PUSH    BX                             ;
        MBOX    35,12,43,30,1                  ;
        MLINE   X,-2,Y,0 ,35,0,16,0 ,1         ;
        MLINE   X,-2,Y,0 ,X ,0,Y ,13,1         ;
        MLINE   X,0 ,Y,13,35,0,21,0 ,1         ;
        MLINE   76,X,Y,0 ,43,0,16,0 ,1         ;
        MLINE   76,X,Y,0 ,78,X,Y ,13,1         ;
        MLINE   78,X,Y,13,43,0,21,0 ,1         ;
        MDAEN   39,10,6,3,1                    ;  パターン作成
        MOV     AX,PITCH                       ;
        ADD     WORD PTR Y,2                   ;
        SUB     X,AX                           ;
        JNE     MKDRT                          ;
        NEG     AX                             ;
        MOV     PITCH,AX                       ;
MKDRT:  POP     BX                             ;
        RET                                    ;
```

```
MKDAT   ENDP

PITCH   DW      2                       ;
X       DW      14                      ;} パターン用ワーク
Y       DW      0                       ;

LINE    PROC
        PUSH    BX                      ;
        PUSH    CX                      ;
        PUSH    DS                      ;
        CALL    GDSET                   ;
        MOV     AX,0                    ;
        MOV     [SI],AX                 ;
        MOV     [SI+2],AX               ;
        MOV     [SI+4],AX               ;
        MOV     AX,CS:LINEX1            ;
        MOV     [SI+6],AX               ;
        MOV     AX,CS:LINEY1            ;
        MOV     [SI+8],AX               ;
        MOV     AX,CS:LINEX2            ;} 直線の描写
        MOV     [SI+0AH],AX             ;
        MOV     AX,CS:LINEY2            ;
        MOV     [SI+0CH],AX             ;
        MOV     AL,0                    ;
        MOV     [SI+0EH],AL             ;
        MOV     AX,CS:COLOR             ;
        MOV     [SI+10H],AX             ;
        MOV     SI,16                   ;
        CALL    GCALL                   ;
        POP     DS                      ;
        POP     CX                      ;
        POP     BX                      ;
        RET                             ;
LINE    ENDP

LINEX1  DW      0                       ;直線の始点のX座標
LINEY1  DW      0                       ;直線の始点のY座標
LINEX2  DW      0                       ;直線の終点のX座標
LINEY2  DW      0                       ;直線の終点のY座標
COLOR   DW      0                       ;直線のカラー

BOX     PROC
        PUSH    BX                      ;
        PUSH    CX                      ;
        PUSH    DS                      ;
        CALL    GDSET                   ;
        MOV     AX,0                    ;
        MOV     [SI],AX                 ;
        MOV     [SI+2],AX               ;
        MOV     [SI+4],AX               ;
        MOV     AX,CS:LINEX1            ;
        MOV     [SI+6],AX               ;
        MOV     AX,CS:LINEY1            ;
        MOV     [SI+8],AX               ;
        MOV     AX,CS:LINEX2            ;
        MOV     [SI+0AH],AX             ;
        MOV     AX,CS:LINEY2            ;} BOX形の描写
        MOV     [SI+0CH],AX             ;
        MOV     AL,0                    ;
        MOV     [SI+0EH],AL             ;
        MOV     AX,CS:COLOR             ;
        MOV     [SI+10H],AX             ;
        MOV     AX,0                    ;
```

```
        MOV     [SI+14H],AX                     ;
        MOV     AL,0                            ;
        MOV     [SI+1CH],AL                     ;
        MOV     SI,18                           ;
        CALL    GCALL                           ;
        POP     DS                              ;
        POP     CX                              ;
        POP     BX                              ;
        RET                                     ;
BOX     ENDP

DAEN    PROC
        PUSH    BX                              ;
        PUSH    CX                              ;
        PUSH    DS                              ;
        CALL    GDSET                           ;
        MOV     AX,0                            ;
        MOV     [SI],AX                         ;
        MOV     [SI+2],AX                       ;
        MOV     [SI+4],AX                       ;
        MOV     AX,CS:DAENX1                    ;
        MOV     [SI+6],AX                       ;
        MOV     AX,CS:DAENY1                    ;
        MOV     [SI+8],AX                       ; 楕円の描写
        MOV     AX,CS:HANKEIX                   ;
        MOV     [SI+0AH],AX                     ;
        MOV     AX,CS:HANKEIY                   ;
        MOV     [SI+0CH],AX                     ;
        MOV     AX,CS:COLOR                     ;
        MOV     WORD PTR [SI+18H],AX            ;
        MOV     AL,0                            ;
        MOV     [SI+24H],AL                     ;
        MOV     AX,CS:COLOR                     ;
        MOV     WORD PTR [SI+26H],AX            ;
        MOV     SI,21                           ;
        CALL    GCALL                           ;
        POP     DS                              ;
        POP     CX                              ;
        POP     BX                              ;
        RET                                     ;
DAEN    ENDP

DAENX1  DW      195                             ;楕円の中心のX座標
DAENY1  DW      60                              ;楕円の中心のY座標
HANKEIX DW      130                             ;X方向半径
HANKEIY DW      60                              ;Y方向半径

TXCLR   LABEL BYTE                              ;テキスト・クリア ＆ 文字の属性の保存
        DB      1BH,"[2J"
        DB      1BH,"[s",1BH,"[>5h",1BH,"[1>h$"

TKEEP   LABEL BYTE                              ;文字の属性の復元
        DB      1BH,"[u",1BH,"[>5l",1BH,"[1>l$"

DAT3D   LABEL BYTE      ;ﾊﾟﾀｰﾝ・ﾃﾞｰﾀ・ｴﾘｱ
        DB      80*10*14*2 DUP(0)
CODE    ENDS

STACK   SEGMENT STACK                           ;スタック・セグメントの定義
        DW      100H    DUP(0)
STACK   ENDS
```

```
BLUE    SEGMENT AT 0A800H                ;B面のセグメントの定義
BLUE    ENDS

RED     SEGMENT AT 0B000H                ;R面のセグメントの定義
RED     ENDS

GREEN   SEGMENT AT 0B800H                ;G面のセグメントの定義
GREEN   ENDS

        END
```

鳥ともマイクのオバケとも取れる妙なものが現れ、テンキーにて自由に動かすことができます。これは単なる線画パターンですが、実際の用途は見ての通り3D（3次元）の画面処理に使われることが多く、それ以外の用途はあまりありません。画面処理の原理は、**図 3-3-1**に示されているようにB面が背景専用、R面とG面は交互に作画（表示オフ）／表示を繰り返すようになっています。つまり、R面が表示中にはG面は作画、G面が表示中にはR面が作画というわけです。このように頻繁にパレット変更をする場合は、『第１章－３』で述べた理由により垂直同期を取らなければなりません。

パレット番号	G面	R面	B面	
0	0	0	0	0（背景色）
1	0	0	1	0（背景色）
2	0	1	0	表示中=4／作画中=0
3	0	1	1	表示中=4／作画中=1
4	1	0	0	作画中=0／表示中=4
5	1	0	1	作画中=1／表示中=4
6	1	1	0	4（常時表示）
7	1	1	1	4（常時表示）

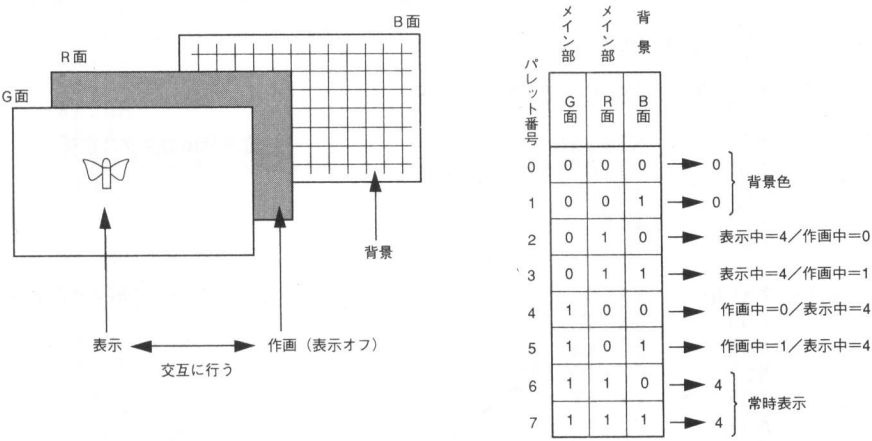



図 3-3-1　３Dの画面処理例

今回、旧パターンを消去するのに画面全体をクリアしていますが、これは3D処理を想定してのものです。というのは、3D画面はサイズが不定のため、消去は表示エリア全体をクリアするか、線を１本ずつ消していくかしかないからです。速度とプログラムの効率を考えれば、普通は前者を採用することになります。

なお、画面処理以外の3Dテクニックには触れていませんが、基本は座標を求める計算式を確立し、２点間をラインで結ぶことができればいいのです。とはいえ、それはそれで一冊の本になるべき大きなテーマです。興味のある方は3Dの専門書を読み、その上でこの節の画面処理テクニックを応用してください。ただし、ゲームのデモ画面などにおける3D表示では、計算後の座標だけをデータとして持っておくと

いう方法も有効です。その場合は、GDCなどによる高速ラインルーチンだけを作成すればOKです。

さて、このプログラムでは背景が固定されていましたが、実際の3D画面では動く背景が必要なこともあります。当然、それは『スカイ・ブルーザー』のような平面スクロール背景などではなく、メイン画面同様3Dで動くものでなければなりません。しかし、これをB面だけで実現しようとすれば、今度は背景に消去／表示のチラつきが現れることになります。そこで、B面にちょっとした細工をし、R面（またはG面）だけで背景とメイン部を表現できるよう工夫をしたのが次のプログラム（**LIST 3-7**）です。

```
A>LIST3-7 ⏎
```

LIST 3-7

```
;*************************************
;*              LIST3-7              *
;*************************************

EXTRN   GTERM:NEAR,GSTAT:NEAR,GMOD8:NEAR,GDSET:NEAR,GCALL:NEAR

CODE    SEGMENT PUBLIC

        ASSUME  CS:CODE,DS:CODE

PRINT   MACRO   STRING                          ;;文字列出力マクロ定義
        MOV     DX,OFFSET STRING
        MOV     AH,09
        INT     21H
        ENDM

VRAMCLS MACRO   VSEG                            ;;VRAMクリア用マクロ定義
        MOV     AX,VSEG
        MOV     ES,AX
        MOV     CX,4000H
        XOR     AX,AX
        MOV     DI,AX
        REP     STOSW
        ENDM

PALSET  MACRO   D1,D2,D3,D4                     ;;パレット操作用マクロ定義
        MOV     AL,D1
        OUT     0AEH,AL
        MOV     AL,D2
        OUT     0AAH,AL
        MOV     AL,D3
        OUT     0ACH,AL
        MOV     AL,D4
        OUT     0A8H,AL
        ENDM

MBOX    MACRO   D1,D2,D3,D4,D5                  ;;BOX描写用マクロ定義
        MOV     WORD PTR COLOR,D5
        MOV     AX,BX
        ADD     AX,D1
        MOV     WORD PTR LINEX1,AX
        MOV     AX,CX
```

```
        ADD     AX,D2
        ADD     AX,AX
        MOV     WORD PTR LINEY1,AX
        MOV     AX,BX
        ADD     AX,D3
        MOV     WORD PTR LINEX2,AX
        MOV     AX,CX
        ADD     AX,D4
        ADD     AX,AX
        MOV     WORD PTR LINEY2,AX
        CALL    BOX
        ENDM

MLINE   MACRO   D1,D2,D3,D4,D5,D6,D7,D8,D9;;直線描写用マクロ定義
        MOV     WORD PTR COLOR,D9
        MOV     AX,BX
        ADD     AX,D1
        SUB     AX,D2
        MOV     WORD PTR LINEX1,AX
        MOV     AX,CX
        ADD     AX,D3
        ADD     AX,D4
        ADD     AX,AX
        MOV     WORD PTR LINEY1,AX
        MOV     AX,BX
        ADD     AX,D5
        SUB     AX,D6
        MOV     WORD PTR LINEX2,AX
        MOV     AX,CX
        ADD     AX,D7
        ADD     AX,D8
        ADD     AX,AX
        MOV     WORD PTR LINEY2,AX
        CALL    LINE
        ENDM

MDAEN   MACRO   D1,D2,D3,D4,D5                  ;;楕円描写用マクロ定義
        MOV     AX,BX
        ADD     AX,D1
        MOV     DAENX1,AX
        MOV     AX,CX
        ADD     AX,D2
        ADD     AX,AX
        MOV     DAENY1,AX
        MOV     HANKEIX,D3
        MOV     HANKEIY,D4
        ADD     HANKEIY,D4
        MOV     AX,D5
        MOV     COLOR,AX
        CALL    DAEN
        ENDM

PMAIN:  CLD
        CALL    GSTAT                           ;グラフィックスの開始
        JNB     $+5                             ;
        JMP     TEXIT                           ;異常終了であればTEXITへ
        CALL    GMOD8                           ;640×400,8色モード・セット
        JNB     $+5                             ;
        JMP     TEXIT                           ;異常終了であればTEXITへ
        PRINT   TXCLR                           ;テキスト画面クリア
        VRAMCLS BLUE                            ;┐
        VRAMCLS RED                             ;┝ グラフィックス画面クリア
        VRAMCLS GREEN                           ;┘
```

```
        PALSET  00H,00H,04H,05H         ;パレットの初期セット
        MOV     CX,8                    ;
        MOV     BX,0                    ;
LP001:  PUSH    CX                      ;
        MOV     CX,0                    ;
        CALL    MKDAT                   ;
        POP     CX                      ;
        ADD     BX,80                   ; パターン作成
        LOOP    LP001                   ;
        MOV     CX,6                    ;
        MOV     BX,0                    ;
LP002:  PUSH    CX                      ;
        MOV     CX,40                   ;
        CALL    MKDAT                   ;
        POP     CX                      ;
        ADD     BX,80                   ;
        LOOP    LP002                   ;
        CALL    MKP3D                   ;パターンデータ作成
        MOV     AX,BLUE                 ;
        MOV     ES,AX                   ;
        MOV     DI,0                    ; B面に背景を作成
        MOV     CX,80*200               ;
        MOV     AX,0AAAAH               ;
        REP     STOSW                   ;
        CALL    DP3GP                   ;G面にパターンを作成し表示（パレット変更）
MLOOP:  CALL    DIRCK                   ;座標移動と方向チェック
        CALL    DP3RP                   ;R面にパターンを作成し表示（パレット変更）
        CALL    DIRCK                   ;座標移動と方向チェック
        CALL    DP3GP                   ;G面にパターンを作成し表示（パレット変更）
        MOV     DL,0FFH                 ;
        MOV     AH,6                    ;
        INT     21H                     ; [ESC] が押されていなければMLOOPへ
        JE      MLOOP                   ;
        CMP     AL,1BH                  ;
        JNE     MLOOP                   ;
        PALSET  04H,15H,26H,37H         ;パレットの状態を初期の状態へ戻す
        PRINT   TKEEP                   ;テキストの属性を復元する
TEXIT:  MOV     AX,4C00H                ;リターンコード・セット
        INT     21H                     ;MS-DOSへ

PAT3D   DB      0                       ;パターン番号
DIR3D   DB      0                       ;パターンの方向
YPO3D   DB      20                      ;パターンのY座標
BAKLL   DB      00000010B               ;背景の左側データ
BAKRR   DB      10000000B               ;背景の右側データ

MKP3D   PROC
        MOV     SI,0                    ;
        MOV     DI,OFFSET DAT3D         ;
        MOV     CX,8                    ;
        CALL    M3DSB                   ; 画面からパターンを作成
        MOV     SI,80*80                ;
        MOV     CX,6                    ;
        CALL    M3DSB                   ;
        RET
MKP3D   ENDP

M3DSB   PROC
        PUSH    DS                      ;
        MOV     AX,BLUE                 ;
        MOV     DS,AX                   ;
        MOV     AX,CS                   ;
        MOV     ES,AX                   ;
```

```
M3DL0:  PUSH    CX                          ;
        PUSH    SI                          ;
        MOV     AX,80                       ;
M3DL1:  MOV     CX,5                        ;  画面からパターンを作成
        REP     MOVSW                       ;
        ADD     SI,80-10                    ;
        DEC     AX                          ;
        JNE     M3DL1                       ;
        POP     SI                          ;
        ADD     SI,10                       ;
        POP     CX                          ;
        LOOP    M3DL0                       ;
        POP     DS                          ;
        RET                                 ;
M3DSB   ENDP

DP3RP   PROC
        MOV     AX,RED                      ;
        CALL    DIP3D                       ;
        CALL    VWAIT                       ;  背景の表示
        PALSET  00H,00H,44H,55H             ;
        RET                                 ;
DP3RP   ENDP

DP3GP   PROC
        MOV     AX,GREEN                    ;
        CALL    DIP3D                       ;
        CALL    VWAIT                       ;  画面クリア
        PALSET  04H,05H,04H,05H             ;
        RET                                 ;
DP3GP   ENDP

DIP3D   PROC
        MOV     OUTPO,AX                    ;
        MOV     ES,AX                       ;
        CALL    BAKMM                       ;
        MOV     AL,PAT3D                    ;
        INC     AL                          ;
        CMP     AL,27                       ;
        JB      PATOK                       ;
        XOR     AL,AL                       ;
PATOK:  MOV     PAT3D,AL                    ;
        CMP     AL,14                       ;
        JB      PTIA1                       ;
        SUB     AL,27                       ;
        NOT     AL                          ;
PTIA1:  INC     AL                          ;
        MOV     BX,OFFSET DAT3D-10*80       ;
        MOV     DX,10*80                    ;
GPADL:  ADD     BX,DX                       ;
        DEC     AL                          ;
        JNE     GPADL                       ;
        PUSH    BX                          ;
        MOV     DX,23H                      ;
        MOV     BX,80-10                    ;
        MOV     AL,DIR3D                    ;
        OR      AL,AL                       ;
        JNE     $+5                         ;
        JMP     DIR30                       ;
        MOV     DX,23H+80*80                ;
        MOV     BX,-80-10                   ;
DIR30:  MOV     DIPON,BX                    ;  R面(またはG面)に背景/パターンを作成し
        MOV     AL,YPO3D                    ;  表示(パレット変更)
```

```
        MOV     BL,AL                   ;
        MOV     BH,0                    ;
        ADD     BX,BX                   ;
        ADD     BX,BX                   ;
        ADD     BX,BX                   ;
        ADD     BX,BX                   ;
        MOV     CH,BH                   ;
        MOV     CL,BL                   ;
        ADD     BX,BX                   ;
        ADD     BX,BX                   ;
        ADD     BX,CX                   ;
        ADD     BX,BX                   ;
        ADD     BX,DX                   ;
        POP     DX                      ;
        MOV     CX,10*100H+80           ;
        MOV     ES,OUTPO                ;
        XCHG    DI,DX                   ;
DP3L0:  PUSH    CX                      ;
DP3L1:  MOV     AL,[DI]                 ;
        OR      ES:[BX],AL              ;
        INC     BX                      ;
        INC     DI                      ;
        DEC     CH                      ;
        JNE     DP3L1                   ;
        ADD     BX,DIPON                ;
        POP     CX                      ;
        DEC     CL                      ;
        JNE     DP3L0                   ;
        XCHG    DI,DX                   ;
        RET                             ;
DIP3D   ENDP

DIPON   DW      80-10                   ;} ワーク・エリア
OUTPO   DW      BLUE                    ;

BAKMM   PROC
        MOV     BX,OFFSET BAKLL         ;
        MOV     AL,DIR3D                ;
        OR      AL,AL                   ;
        JNE     ROTA1                   ;
        ROL     BYTE PTR [BX],1         ;
        ROL     BYTE PTR [BX],1         ;
        MOV     DH,[BX]                 ;
        INC     BX                      ;
        ROR     BYTE PTR [BX],1         ;
        ROR     BYTE PTR [BX],1         ;
        JMP     BAKM1                   ;
ROTA1:  ROR     BYTE PTR [BX],1         ;
        ROR     BYTE PTR [BX],1         ;
        MOV     DH,[BX]                 ;
        INC     BX                      ;
        ROL     BYTE PTR [BX],1         ;
        ROL     BYTE PTR [BX],1         ;
BAKM1:  MOV     DL,[BX]                 ;
        MOV     BX,10   ;VRAM ADDRES    ;
        MOV     CX,400                  ;
MBALP:  PUSH    CX                      ;
        MOV     AL,DH                   ;
        OR      AL,10100000B            ;
        MOV     ES:[BX],AL              ;
        INC     BX                      ;
        MOV     CH,27                   ;
MBAL0:  MOV     ES:[BX],DH              ;
        INC     BX                      ;} 背景をローテートして指定プレーンに表示
```

```
        DEC     CH                          ;
        JNE     MBAL0                       ;
        MOV     AL,DH                       ;
        OR      AL,00001010B                ;
        MOV     ES:[BX],AL                  ;
        INC     BX                          ;
        MOV     BYTE PTR ES:[BX],0          ;
        INC     BX                          ;
        MOV     BYTE PTR ES:[BX],0          ;
        INC     BX                          ;
        MOV     AL,DL                       ;
        OR      AL,10100000B                ;
        MOV     ES:[BX],AL                  ;
        INC     BX                          ;
RBVAL:  MOV     CH,27                       ;
MBAL1:  MOV     ES:[BX],DL                  ;
        INC     BX                          ;
        DEC     CH                          ;
        JNE     MBAL1                       ;
        MOV     AL,DL                       ;
        OR      AL,00001010B                ;
        MOV     ES:[BX],AL                  ;
        ADD     BX,21                       ;
        POP     CX                          ;
        LOOP    MBALP                       ;
        RET
BAKMM   ENDP

VWAIT   PROC
        IN      AL,0A0H                     ;
        TEST    AL,00100000B                ;
        JNE     VWAIT                       ;  ウェイト・ルーチン
VWAT2:  IN      AL,0A0H                     ;
        TEST    AL,00100000B                ;
        JE      VWAT2                       ;
        RET                                 ;
VWAIT   ENDP

DIRCK   PROC
        MOV     BX,OFFSET YP03D             ;
        MOV     AL,DIR3D                    ;
        OR      AL,AL                       ;
        JE      DIR00                       ;
        MOV     AL,[BX]                     ;
        INC     AL                          ;
        CMP     AL,161                      ;
        JE      DIRRV                       ;  パターン座標を変更し移動方向をチェック
        MOV     [BX],AL                     ;
        JMP     DIRRT                       ;
DIR00:  MOV     AL,[BX]                     ;
        SUB     AL,1                        ;
        JB      DIRRV                       ;
        MOV     [BX],AL                     ;
        JMP     DIRRT                       ;
DIRRV:  XOR     BYTE PTR DIR3D,1            ;
DIRRT:  RET                                 ;
DIRCK   ENDP

MKDAT   PROC
        PUSH    BX                          ;
        MBOX    35,12,43,30,1               ;
        MBOX    36,12,44,30,1               ;
        MLINE   X,-2,Y,0 ,35,0,16,0 ,1      ;
```

```
        MLINE   X,-2,Y,0 ,X ,0,Y ,13,1     ;
        MLINE   X,0 ,Y,13,35,0,21,0 ,1     ;
        MLINE   76,X,Y,0 ,43,0,16,0 ,1     ;
        MLINE   76,X,Y,0 ,78,X,Y ,13,1     ;
        MLINE   78,X,Y,13,43,0,21,0 ,1     ;
        INC     X                          ;
        MLINE   X,-2,Y,0 ,35,0,16,0 ,1     ;
        MLINE   X,-2,Y,0 ,X ,0,Y ,13,1     ;
        MLINE   X,0 ,Y,13,35,0,21,0 ,1     ;
        MLINE   76,X,Y,0 ,43,0,16,0 ,1     ;  パターンの作成
        MLINE   76,X,Y,0 ,78,X,Y ,13,1     ;
        MLINE   78,X,Y,13,43,0,21,0 ,1     ;
        DEC     X                          ;
        MDAEN   39,10,6,3,1                ;
        MDAEN   40,10,6,3,1                ;
        MOV     AX,PITCH                   ;
        ADD     WORD PTR Y,2               ;
        SUB     X,AX                       ;
        JNE     MKDRT                      ;
        NEG     AX                         ;
        MOV     PITCH,AX                   ;
MKDRT:  POP     BX                         ;
        RET                                ;
MKDAT   ENDP

PITCH   DW      2                          ;
X       DW      14                         ;  ワーク・エリア
Y       DW      0                          ;

LINE    PROC
        PUSH    BX                         ;
        PUSH    CX                         ;
        PUSH    DS                         ;
        CALL    GDSET                      ;
        MOV     AX,0                       ;
        MOV     [SI],AX                    ;
        MOV     [SI+2],AX                  ;
        MOV     [SI+4],AX                  ;
        MOV     AX,CS:LINEX1               ;
        MOV     [SI+6],AX                  ;
        MOV     AX,CS:LINEY1               ;
        MOV     [SI+8],AX                  ;
        MOV     AX,CS:LINEX2               ;  直線の描写
        MOV     [SI+0AH],AX                ;
        MOV     AX,CS:LINEY2               ;
        MOV     [SI+0CH],AX                ;
        MOV     AL,0                       ;
        MOV     [SI+0EH],AL                ;
        MOV     AX,CS:COLOR                ;
        MOV     [SI+10H],AX                ;
        MOV     AX,1                       ;
        MOV     [SI+14H],AX                ;
        MOV     SI,16                      ;
        CALL    GCALL                      ;
        POP     DS                         ;
        POP     CX                         ;
        POP     BX                         ;
        RET                                ;
LINE    ENDP

LINEX1  DW      0                          ;直線の始点のX座標
LINEY1  DW      0                          ;直線の始点のY座標
LINEX2  DW      0                          ;直線の終点のX座標
```

```
LINEY2  DW      0                           ;直線の終点のY座標
COLOR   DW      0                           ;直線の色

BOX     PROC
        PUSH    BX                          ;
        PUSH    CX                          ;
        PUSH    DS                          ;
        CALL    GDSET                       ;
        MOV     AX,0                        ;
        MOV     [SI],AX                     ;
        MOV     [SI+2],AX                   ;
        MOV     [SI+4],AX                   ;
        MOV     AX,CS:LINEX1                ;
        MOV     [SI+6],AX                   ;
        MOV     AX,CS:LINEY1                ;
        MOV     [SI+8],AX                   ;
        MOV     AX,CS:LINEX2                ;
        MOV     [SI+0AH],AX                 ;
        MOV     AX,CS:LINEY2                ; BOX形の描写
        MOV     [SI+0CH],AX                 ;
        MOV     AL,0                        ;
        MOV     [SI+0EH],AL                 ;
        MOV     AX,CS:COLOR                 ;
        MOV     [SI+10H],AX                 ;
        MOV     AX,1                        ;
        MOV     [SI+14H],AX                 ;
        MOV     AL,0                        ;
        MOV     [SI+1CH],AL                 ;
        MOV     SI,18                       ;
        CALL    GCALL                       ;
        POP     DS                          ;
        POP     CX                          ;
        POP     BX                          ;
        RET                                 ;
BOX     ENDP

DAEN    PROC
        PUSH    BX                          ;
        PUSH    CX                          ;
        PUSH    DS                          ;
        CALL    GDSET                       ;
        MOV     AX,0                        ;
        MOV     [SI],AX                     ;
        MOV     [SI+2],AX                   ;
        MOV     [SI+4],AX                   ;
        MOV     AX,CS:DAENX1                ;
        MOV     [SI+6],AX                   ;
        MOV     AX,CS:DAENY1                ;
        MOV     [SI+8],AX                   ; 楕円の描写
        MOV     AX,CS:HANKEIX               ;
        MOV     [SI+0AH],AX                 ;
        MOV     AX,CS:HANKEIY               ;
        MOV     [SI+0CH],AX                 ;
        MOV     AX,CS:COLOR                 ;
        MOV     WORD PTR [SI+18H],AX        ;
        MOV     AL,0                        ;
        MOV     [SI+24H],AL                 ;
        MOV     AX,CS:COLOR                 ;
        MOV     WORD PTR [SI+26H],AX        ;
        MOV     SI,21                       ;
        CALL    GCALL                       ;
        POP     DS                          ;
        POP     CX                          ;
```

```
        POP     BX                          ;┐
        RET                                 ;┘
DAEN    ENDP

DAENX1  DW      195                         ;楕円の中心のX座標
DAENY1  DW      60                          ;楕円の中心のY座標
HANKEIX DW      130                         ;X方向半径
HANKEIY DW      60                          ;Y方向半径

TXCLR   LABEL BYTE                          ;テキスト・クリア　&　文字の属性の保存
        DB      1BH,"[2J"
        DB      1BH,"[s",1BH,"[>5h",1BH,"[1>h$"

TKEEP   LABEL BYTE                          ;文字の属性の復元
        DB      1BH,"[u",1BH,"[>5l",1BH,"[1>l$"

DAT3D   LABEL BYTE                          ;パターンデータエリア
        DB      80*10*14*2 DUP(0)

CODE    ENDS

STACK   SEGMENT STACK                       ;スタック・セグメントの定義
        DW      100H    DUP(0)
STACK   ENDS

BLUE    SEGMENT AT 0A800H                   ;B面のセグメントの定義
BLUE    ENDS

RED     SEGMENT AT 0B000H                   ;R面のセグメントの定義
RED     ENDS

GREEN   SEGMENT AT 0B800H                   ;G面のセグメントの定義
GREEN   ENDS

        END
```

別々の色で背景とメイン部が表示されていますが、これらはどちらも同じプレーンに描かれたものです。よく見ると、背景はカラー５／０のタイリング、メイン部はカラー５／４のタイリングで描かれています。実は、このタイリングカラーを実現している秘密がＢ面にあるのです。では、ESC キーでプログラムの実行を停止してください。

パレットが初期化され、B面はすべて 10101010Bで埋め尽くされていることがわかります。これがB面の秘密です。こうしておくと、R面（またはG面）にデータをストアすればカラー５／４のタイリングとすることができるし、データを 10101010Bと ANDを取ってストアすれば、カラー５／０のタイリングとすることができます(図 3-3-2参照)。

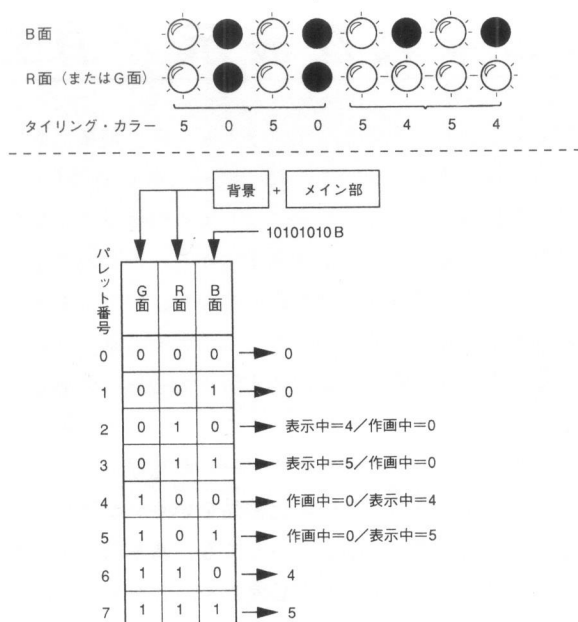
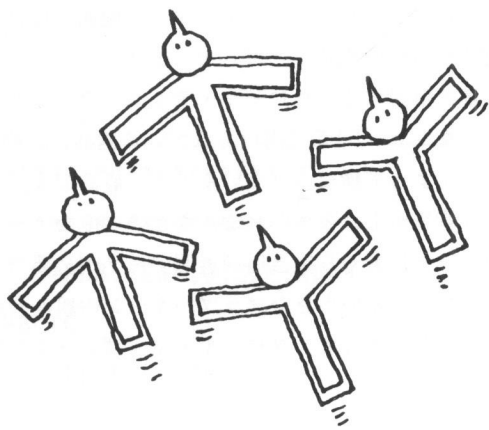



図 3-3-2　３Ｄタイリング・カラー

ここではテスト的な背景ですから、特に背景表示の際に ANDを取るという作業もありませんが、実際には背景はすべて 10101010Bと ANDを取り、メイン部をORで重ね合わせるという手順となります。その際、メイン部のラインは必ず横２ドット以上で構成しなければならないということに注意してください。こうしないと、条件によっては背景と同化してしまうことがあるからです。

さて、このプログラムにはテンキー操作を省いた代わりに、パターンの上下反転表示を取り入れてあります。上から下へ表示するか、下から上へ表示するか、たったそれだけの違いですが、データ活用のミニ・テクニックとして覚えておくと便利です。

3－4　おしゃれな表示／消去

出会いと別れ、入学と卒業、生と死……。何事においても、始まりがあるものには終わりがあります。そして、ヒーロー（あるいはヒロイン）と呼ばれる人たちは、それが実にカッコイイのです。ましてや映画やテレビなどでは、現実と違いドジや失敗があってもやり直しがききますから、われわれがマネをしようと思ってもそうウマクいくものではありません。しかし、現実はダメでもやり直しのきくプログラムの世界でなら、いくらでもカッコよさを追求できるではありませんか。

画面に表示されたものは、必ず消え去る運命にあります。ゲームも一種のエンターテインメントの世界ですから、ここにカッコよさを取り入れるのは制作者の務めです。もしかすると、それだけでゲームが見違えるようになるかもしれません。この節では、そんな表示／消去の化粧品をいくつか集めてみました。

さて、一口に表示／消去といっても、基本的にデータの不要な消去に対し、表示にはグラフィック・データが必要です。しかも、それはゲームの顔であるタイトル画面、アドベンチャー・ゲームなどのシーン画面、パターンによるマップ画面……等、状況に応じてデータ構造が違います。例えば、タイトルやエンディング画面では、表示過程を特に効果的に見せるために大量のデータをベタで持つこともありますが、アドベンチャー・ゲームのシーン画面ではデータに圧縮をかけるのが普通です。また、パターンによるマップ画面ではパターンのサイズやマップデータの構造も問題となるでしょう。

グラフィック・データの圧縮／展開については、テーマとして大き過ぎるだけでなく展開が圧縮の逆の作業であることを考えると化粧にも限界があります。そのため、1節で取り上げたテキスト画面によるブラインド処理が最適の化粧品となっています。となると残るは2つですが、この2つのグラフィック・データはサイズが違うだけで基本的には同じ構造（B／R／G順に並んだもの）です。そこで、具体的な化粧に入る前に、まずパターンによるマップ画面とはどのようなものなのか、マップデータの構造から確認することにしましょう。

これは次の章で解説するスクロール・テクニックにも関連していることなのですが、いわゆるアクションRPGで用いられている画面は、横16ドット×縦16ドットのマップパターンによって構成されているのが一般的です。逆重ね合わせ（部分的にキャラクタより背景が上に表示される）などの高度なテクニックを使う場合は、マップパターンのデータにもくり抜き用の透明データが必要ですが、ここではそれは考えないことにします。この小さなマップパターンの組み合わせがマップ画面となり、それが連続することによってスクロール・マップデータとなるわけです（図 3-4-1)。

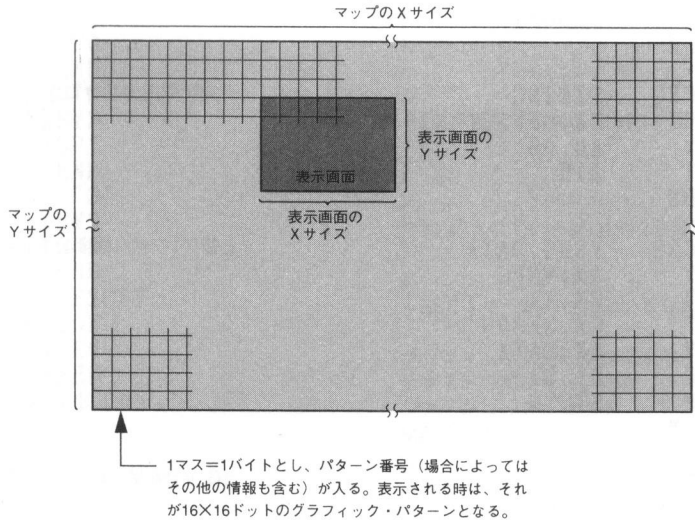



図 3-4-1　マップデータの構造

スクロールの手法はともかく、スタートはマップデータの一部を画面に表示することです。これはなにもゲームのスタート時だけではなく、町やフィールド、ダンジョンとの出入りなど、画面が切り替わるたびに必要になることです。最もシンプルなのは、画面上部から順に表示していく方法ですが、これは表示速度が速くプログラムも簡単ということ以外に魅力はありません。もちろん、このことも画面表示の重要なファクターですが、今回はここにカッコよさを求めているわけですから、それ以上に「見せる」という工夫が要求されています。

そこで、内容はあくまでもテスト的なものですが、サイズフリーのマップとサイズフリーの表示画面に対応したものを２組、そしてタイトルなどベタ・データに対応したものを１組、計３組の化粧プログラムを用意してみました。では、プログラム（LIST 3-8）を実行してください。

```
A>LIST3-8 ⏎
```

LIST 3-8

```
;************************************
;*              LIST3-8             *
;************************************

EXTRN   GSTAT:NEAR,GMOD8:NEAR,GTERM:NEAR

CODE    SEGMENT PUBLIC

        ASSUME  CS:CODE,DS:CODE

GETKEY  MACRO                                    ;;1文字入力マクロ定義
        LOCAL   GLOOP
GLOOP:  MOV     DL,0FFH
        MOV     AH,6
        INT     21H
```

```
        JZ      GLOOP
        ENDM

PRINT   MACRO   STRING                          ;;文字列出力マクロ定義
        MOV     DX,OFFSET STRING
        MOV     AH,09
        INT     21H
        ENDM

VFILL   MACRO   VSEG,DATA                       ;;各プレーン1画面分をDATAで埋める
        MOV     AX,VSEG
        MOV     ES,AX
        MOV     CX,3E80H
        MOV     AX,DATA
        XOR     DI,DI
        REP     STOSW
        ENDM

VRAMW1  MACRO   VSEG                            ;;G.VRAMへのデータ展開マクロ定義
        LOCAL   SDIL2
        PUSH    DI
        MOV     AX,VSEG
        MOV     ES,AX
        MOV     CX,8
SDIL2:  MOVSW
        SUB     SI,2
        ADD     DI,80-2
        MOVSW
        ADD     DI,80-2
        LOOP    SDIL2
        POP     DI
        ENDM

ESCKY   EQU     1BH                             ; ESC キーのアスキーコード

PMAIN:  CLD
        CALL    GSTAT                           ;グラフィックスの開始
        JNB     $+5                             ;
        JMP     TEXIT                           ;異常終了であればTEXITへ
        CALL    GMOD8                           ;640×400,8色モード・セット
        JNB     $+5                             ;
        JMP     TEXIT                           ;異常終了あればTEXITへ
        VFILL   BLUE ,0AAAAH                    ;
        VFILL   RED  ,0                         ;  背景の表示
        VFILL   GREEN,0AAAAH                    ;
        PRINT   TXCLR                           ;テキスト画面クリア
        XOR     AL,AL                           ;AL←0
        MOV     DCEDT,AL                        ;消去／表示フラグを初期化
        CALL    SDISP                           ;初期画面表示
        MOV     DI,OFFSET KPMDT                 ;
        PUSH    DS                              ;
        MOV     AX,CS                           ;
        MOV     ES,AX                           ;
        MOV     AX,BLUE                         ;  初期画面をKPMDTへデータ化
        CALL    MDATA                           ;
        MOV     AX,RED                          ;
        CALL    MDATA                           ;
        MOV     AX,GREEN                        ;
        CALL    MDATA                           ;
        POP     DS                              ;
DCEND:  XOR     BYTE PTR DCEDT,1                ;消去／表示フラグを反転
MLOOP:  GETKEY                                  ;1文字入力
        CMP     AL,"1"                          ;;1
```

```
        JNE     NKEY1                          ;} 1が押されていればDISP0を実行
        CALL    DISP0                          ;
        JMP     DCEND                          ;
NKEY1:  CMP     AL,"2"                         ;
        JNE     NKEY2                          ;} 2が押されていればDISP1を実行
        CALL    DISP1                          ;
        JMP     DCEND                          ;
NKEY2:  CMP     AL,"3"                         ;
        JNE     NKEY3                          ;} 3が押されていればDISP2を実行
        CALL    DISP2                          ;
        JMP     DCEND                          ;
NKEY3:  CMP     AL,ESCKY                       ; [ESC] キーが押されていなければMLOOPへ
        JNE     MLOOP                          ;
        CALL    GTERM                          ;グラフィックス処理を終了する
        PRINT   TKEEP                          ;テキストの属性を元に戻す
TEXIT:  MOV     AX,4C00H                       ;リターンコード・セット
        INT     21H                            ;MS-DOSへ

DCEDT   DB      0                              ;消去／表示フラグ

MDATA   PROC
        MOV     DS,AX                          ;
        MOV     SI,DVRAM                       ;
        MOV     AX,GYMAX*16                    ;
MDATL:  MOV     CX,GXMAX                       ;
        REP     MOVSW                          ;} プレーン別に表示画面のデータを転送
        ADD     SI,80-GXMAX*2                  ;
        DEC     AX                             ;
        JNE     MDATL                          ;
        RET                                    ;
MDATA   ENDP

VWCHK   PROC
        MOV     AX,40EH                        ;
        INT     18H                            ;
        TEST    AH,1H                          ;
        JE      VWRET                          ;
VWAIT:  IN      AL,0A0H                        ;
        TEST    AL,00100000B                   ;
        JNE     VWAIT                          ;
VWAT2:  IN      AL,0A0H                        ;} [SHIFT] が押されていればウェイトを取る
        TEST    AL,00100000B                   ;
        JE      VWAT2                          ;
        PUSH    CX                             ;
        MOV     CX,2000H                       ;
WAIT0:  LOOP    WAIT0                          ;
        POP     CX                             ;
VWRET:  RET                                    ;
VWCHK   ENDP

                                               ;表示開始アドレス
DVRAM   EQU     0A10H                          ;表示パターン数（X）
GXMAX   EQU     24                             ;表示パターン数（Y）
GYMAX   EQU     17                             ;表示画面ベタデータ数（プレーン別）
G1VAL   EQU     GXMAX*2*GYMAX*16

MXMAX   EQU     24                             ;マップサイズ（X）
MYMAX   EQU     17                             ;マップサイズ（Y）
MTOPX   DB      0                              ;表示マップトップ（X）
MTOPY   DB      0                              ;表示マップトップ（Y）

SDISP   PROC
        MOV     CX,0                           ;
        CALL    MAPAD                          ;
```

```
        MOV     BP,BX                   ;
        MOV     DI,DVRAM                ;
        MOV     CL,GYMAX                ;
SDIL0:  MOV     CH,GXMAX                ;
SDIL1:  PUSH    CX                      ;
        MOV     AL,CS:[BP]              ;
        INC     BP                      ;
        CALL    PTADR                   ;
        VRAMW1  BLUE                    ;
        VRAMW1  RED                     ; マップにしたがって画面に表示する（ノーマル）
        VRAMW1  GREEN                   ;
        INC     DI                      ;
        INC     DI                      ;
        POP     CX                      ;
        DEC     CH                      ;
        JNE     SDIL1                   ;
        ADD     DI,80*16-GXMAX*2        ;
        ADD     BP,MXMAX-GXMAX          ;
        DEC     CL                      ;
        JNE     SDIL0                   ;
        RET                             ;
SDISP   ENDP

DISP0   PROC
        MOV     CX,0                    ;
        MOV     DH,GYMAX-1              ;
        MOV     DL,GXMAX                ;
        JMP     DIP0L                   ;
DIP00:  PUSH    DX                      ;
YDOUL:  CALL    BCDP0                   ;
        INC     CL                      ;
        DEC     DL                      ; 画面の上側を表示する
        JNE     YDOUL                   ;
        DEC     CL                      ;
        INC     CH                      ;
        POP     DX                      ;
        DEC     DL                      ;
        OR      DH,DH                   ;
        JE      DIPRT                   ;
        PUSH    DX                      ;
YDORL:  CALL    BCDP0                   ;
        INC     CH                      ;
        DEC     DH                      ;
        JNE     YDORL                   ;
        DEC     CH                      ; 画面の右側を表示する
        DEC     CL                      ;
        POP     DX                      ;
        DEC     DH                      ;
        OR      DL,DL                   ;
        JE      DIPRT                   ;
        PUSH    DX                      ;
YDODL:  CALL    BCDP0                   ;
        DEC     CL                      ;
        DEC     DL                      ;
        JNE     YDODL                   ;
        INC     CL                      ; 画面の下側を表示する
        DEC     CH                      ;
        POP     DX                      ;
        DEC     DL                      ;
        OR      DH,DH                   ;
        JE      DIPRT                   ;
        PUSH    DX                      ;
YDOLL:  CALL    BCDP0                   ;
```

```
        DEC     CH                      ;
        DEC     DH                      ;
        JNE     YDOLL                   ;  画面の左側を表示する
        INC     CH                      ;
        INC     CL                      ;
        POP     DX                      ;
        DEC     DH                      ;
DIPOL:  OR      DL,DL                   ;
        JE      DIPRT                   ;
        JMP     DIP00                   ;DIP00へ
DIPRT:  RET                             ;リターン
DISP0   ENDP

BCDP0   PROC
        PUSH    CX                      ;
        PUSH    DX                      ;
        CALL    MAPAD                   ;
        MOV     AL,[BX]                 ;
        CALL    PTADR                   ;
        CALL    BCADR                   ;
        MOV     AX,BLUE                 ;
        CALL    BC0SB                   ;
        MOV     AX,RED                  ;
        CALL    BC0SB                   ;
        MOV     AX,GREEN                ;  (CL,CH)の位置にマップを表示（消去）する
        CALL    BC0SB                   ;
        MOV     CL,30                   ;
BWLOP:  DEC     CH                      ;
        JNE     BWLOP                   ;
        DEC     CL                      ;
        JNE     BWLOP                   ;
        CALL    VWCHK                   ;
        POP     DX                      ;
        POP     CX                      ;
        RET                             ;
BCDP0   ENDP

BC0SB   PROC
        PUSH    DI                      ;
        PUSH    DS                      ;
        MOV     ES,AX                   ;  消去／表示のフラグチェック
        MOV     AL,DCEDT                ;
        OR      AL,AL                   ;
        JNE     B0CLS                   ;
        MOV     CX,8                    ;
BC0SL:  MOVSW                           ;
        SUB     SI,2                    ;
        ADD     DI,80-2                 ;  マップ表示のサブルーチン
        MOVSW                           ;
        ADD     DI,80-2                 ;
        LOOP    BC0SL                   ;
        JMP     BC0RT                   ;
B0CLS:  XOR     AX,AX                   ;
        MOV     CX,16                   ;
B0CLP:  MOV     ES:[DI],AX              ;
        ADD     DI,80                   ;  画面上のマップ消去サブルーチン
        LOOP    B0CLP                   ;
BC0RT:  POP     DS                      ;
        POP     DI                      ;
        RET
BC0SB   ENDP

DISP1   PROC
```

```
        MOV     AL,DCEDT               ;]
        OR      AL,AL                  ;} 消去／表示のフラグチェック
        JE      DISPB                  ;]
        CALL    D1CLS                  ;消去ルーチンコール
        JMP     DIS1RT                 ;リターン
DISPB:  MOV     CX,0                   ;]
        CALL    MAPAD                  ;
        MOV     BP,BX                  ;
        MOV     DX,MXMAX-GXMAX         ;
        MOV     CX,2*16                ;
        MOV     SI,OFFSET V1TOP        ;
D1LOP:  PUSH    CX                     ;} 画面上にマップをジワジワと表示する
        PUSH    BP                     ;
        CALL    D1SUB                  ;
        CALL    VWAIT                  ;
        POP     BP                     ;
        POP     CX                     ;
        LOOP    D1LOP                  ;
DIS1RT: RET                            ;]
DISP1   ENDP

D1PON   DB      0,0

D1SUB   PROC
        MOV     AL,[SI+0]              ;]
        MOV     D1PON,AL               ;
        MOV     BX,[SI+1]              ;
        ADD     SI,3                   ;
        PUSH    SI                     ;
        MOV     CL,GYMAX               ;
D1LP0:  MOV     CH,GXMAX               ;
D1LP1:  MOV     AL,CS:[BP]             ;
        INC     BP                     ;
        CALL    PTADR                  ;
        ADD     SI,WORD PTR D1PON      ;
        MOV     AX,BLUE                ;
        MOV     ES,AX                  ;
        MOV     AL,[SI+0]              ;
        MOV     ES:[BX],AL             ;
        MOV     AX,RED                 ;} 1バイトを一定のアルゴリズムで表示
        MOV     ES,AX                  ;  CL＝表示パターン数（Y）
        MOV     AL,[SI+16]             ;  CH＝表示パターン数（X）
        MOV     ES:[BX],AL             ;  BP＝マップアドレス
        MOV     AX,GREEN               ;  BX表示アドレス
        MOV     ES,AX                  ;
        MOV     AL,[SI+32]             ;
        MOV     ES:[BX],AL             ;
        INC     BX                     ;
        INC     BX                     ;
        DEC     CH                     ;
        JNE     D1LP1                  ;
        ADD     BX,80*16-GXMAX*2       ;
        ADD     BP,MXMAX-GXMAX         ;
        CALL    VWCHK                  ;
        DEC     CL                     ;
        JNE     D1LP0                  ;
        POP     SI                     ;
        RET                            ;]
D1SUB   ENDP

V1TOP   LABEL BYTE                     ;1バイトの表示位置情報
        DB      0
        DW      DVRAM
```

```
        DB      5
        DW      DVRAM+160*2+1
        DB      10
        DW      DVRAM+160*5
        DB      15
        DW      DVRAM+160*7+1
        DB      4
        DW      DVRAM+160*2
        DB      9
        DW      DVRAM+160*4+1
        DB      14
        DW      DVRAM+160*7
        DB      3
        DW      DVRAM+160+1
        DB      8
        DW      DVRAM+160*4
        DB      13
        DW      DVRAM+160*6+1
        DB      2
        DW      DVRAM+160
        DB      7
        DW      DVRAM+160*3+1
        DB      12
        DW      DVRAM+160*6
        DB      1
        DW      DVRAM+1
        DB      6
        DW      DVRAM+160*3
        DB      11
        DW      DVRAM+160*5+1
        DB      0
        DW      DVRAM+80
        DB      5
        DW      DVRAM+80+160*2+1
        DB      10
        DW      DVRAM+80+160*5
        DB      15
        DW      DVRAM+80+160*7+1
        DB      4
        DW      DVRAM+80+160*2
        DB      9
        DW      DVRAM+80+160*4+1
        DB      14
        DW      DVRAM+80+160*7
        DB      3
        DW      DVRAM+80+160+1
        DB      8
        DW      DVRAM+80+160*4
        DB      13
        DW      DVRAM+80+160*6+1
        DB      2
        DW      DVRAM+80+160
        DB      7
        DW      DVRAM+80+160*3+1
        DB      12
        DW      DVRAM+80+160*6
        DB      1
        DW      DVRAM+80+1
        DB      6
        DW      DVRAM+80+160*3
        DB      11
        DW      DVRAM+80+160*5+1
```

```
D1CLS   PROC
        MOV     DL,01110111B            ;
        MOV     DH,11101110B            ;
        CALL    C1SUB                   ;
        CALL    C1SUB                   ;
        CALL    C1SUB                   ;
C1SUB:  MOV     BX,DVRAM                ;
        CALL    C1SB2                   ; 表示画面をジワジワと消去
        ROR     DL,1                    ;
        XCHG    DH,DL                   ;
        MOV     BX,DVRAM+80             ;
        CALL    C1SB2                   ;
        ROL     DL,1                    ;
        XCHG    DH,DL                   ;
        RET                             ;
D1CLS   ENDP

C1SB2   PROC
        PUSH    DS                      ;
        MOV     BP,GYMAX*8              ;
C1SL0:  MOV     CX,GXMAX*2              ;
C1SL1:  MOV     AX,BLUE                 ;
        MOV     DS,AX                   ;
        AND     [BX],DL                 ;
        MOV     AX,RED                  ;
        MOV     DS,AX                   ;
        AND     [BX],DL                 ;
        MOV     AX,GREEN                ; ドット単位で表示画面を消去する
        MOV     DS,AX                   ;
        AND     [BX],DL                 ;
        INC     BX                      ;
        LOOP    C1SL1                   ;
        CALL    VWCHK                   ;
        ADD     BX,80*2-GXMAX*2         ;
        DEC     BP                      ;
        JNE     C1SL0                   ;
        POP     DS                      ;
        RET                             ;
C1SB2   ENDP

DISP2   PROC
        MOV     AL,DCEDT                ;
        OR      AL,AL                   ; 消去／表示のフラグチェック
        JNE     D2CLS                   ;
        MOV     BX,OFFSET KPMDT - GXMAX*2               ;
        MOV     AX,BLUE                                 ;
        CALL    D2SUB                                   ;
        MOV     BX,OFFSET KPMDT + G1VAL - GXMAX*2       ;
        MOV     AX,RED                                  ; 画面上にマップを斜めに表示
        CALL    D2SUB                                   ;
        MOV     BX,OFFSET KPMDT + G1VAL*2 - GXMAX*2     ;
        MOV     AX,GREEN                                ;
        CALL    D2SUB                                   ;
        RET                                             ;
DISP2   ENDP

D2SUB   PROC
        MOV     ES,AX                   ;
        MOV     GHPON,BX                ;
        MOV     CX,0                    ;
        MOV     BP,0                    ;
        CALL    GETHD                   ;
        MOV     DX,G1VAL                ;
```

```
D2SLP:  PUSH    DX                      ;
        MOV     AL,[SI]                 ;
        MOV     ES:[BX],AL              ;
        ADD     SI,GXMAX*2+1            ;
        ADD     BX,80+1                 ;
        INC     CX                      ;
        CMP     CX,GXMAX*2              ; プレーン別に斜め表示を行う
        JNE     D2COK                   ;
        MOV     CX,0                    ;
        JMP     CGTHD                   ;
D2COK:  INC     BP                      ;
        CMP     BP,GYMAX*16             ;
        JNE     D2BOK                   ;
        MOV     BP,0                    ;
CGTHD:  CALL    GETHD                   ;
        CALL    VWCHK                   ;
D2BOK:  POP     DX                      ;
        DEC     DX                      ;
        JNE     D2SLP                   ;
        RET                             ;
D2SUB   ENDP

D2CLS   PROC
        MOV     AX,GREEN                ;
        CALL    C2SUB                   ;
        MOV     AX,RED                  ;
        CALL    C2SUB                   ; 表示画面を斜め消去
        MOV     AX,BLUE                 ;
        CALL    C2SUB                   ;
        RET                             ;
D2CLS   ENDP

C2SUB   PROC
        MOV     ES,AX                   ;
        MOV     CX,0                    ;
        MOV     BP,0                    ;
        CALL    V2ADR                   ;
        MOV     DX,G1VAL                ;
C2SLP:  PUSH    DX                      ;
        MOV     BYTE PTR ES:[BX],0      ;
        MOV     DX,80+1                 ;
        ADD     BX,DX                   ;
        INC     CX                      ;
        CMP     CX,GXMAX*2              ;
        JNE     C2COK                   ; プレーン別に斜め消去を行う
        MOV     CX,0                    ;
        JMP     CV2AD                   ;
C2COK:  INC     BP                      ;
        CMP     BP,GYMAX*16             ;
        JNE     C2BOK                   ;
        MOV     BP,0                    ;
CV2AD:  CALL    V2ADR                   ;
        CALL    VWCHK                   ;
C2BOK:  POP     DX                      ;
        DEC     DX                      ;
        JNE     C2SLP                   ;
        RET                             ;
C2SUB   ENDP

GHPON   DW      OFFSET KPMDT - GXMAX*2

GETHD   PROC
        MOV     AX,BP                   ;
```

```
        MOV     SI,GHPON                ;
        MOV     DX,GXMAX*2              ;
        MUL     DL                      ;
        ADD     SI,AX                   ;
        ADD     SI,CX                   ; (CL,CH) からSIにデータアドレス、BXに
V2ADR:  MOV     AX,80                   ; 表示アドレスを求める
        MUL     BP                      ;
        MOV     BX,AX                   ;
        ADD     BX,DVRAM                ;
        ADD     BX,CX                   ;
        RET                             ;
GETHD   ENDP

MAPAD   PROC
        PUSH    CX                      ;
        MOV     AL,MTOPX                ;
        ADD     AL,CL                   ;
        MOV     CL,AL                   ;
        MOV     AL,MTOPY                ;
        ADD     AL,CH                   ;
        MOV     CH,AL                   ;
        MOV     BX,OFFSET GDTOP-MXMAX   ; (CL,CH) からBXにマップデータ・アドレス
        MOV     DX,MXMAX                ; を求める
        INC     CH                      ;
MNOLP:  ADD     BX,DX                   ;
        DEC     CH                      ;
        JNE     MNOLP                   ;
        ADD     BX,CX                   ;
        POP     CX                      ;
        RET                             ;
MAPAD   ENDP

PTADR   PROC
        MOV     AH,AL                   ;
        ROL     AL,1                    ;
        ADD     AL,AH                   ;
        MOV     AH,0                    ;
        ADD     AX,AX                   ;
        ADD     AX,AX                   ; ALからSIにパターンデータ・アドレスを求める
        ADD     AX,AX                   ;
        ADD     AX,AX                   ;
        MOV     SI,OFFSET GDATA         ;
        ADD     SI,AX                   ;
        RET                             ;
PTADR   ENDP

BCADR   PROC
        MOV     BX,DVRAM-80*16          ;
        MOV     DX,80*16                ;
        INC     CH                      ;
BCAL0:  ADD     BX,DX                   ;
        DEC     CH                      ; (CL,CH) からDIへ表示アドレスを求める
        JNE     BCAL0                   ;
        SAL     CL,1                    ;
        ADD     BX,CX                   ;
        MOV     DI,BX                   ;
        RET                             ;
BCADR   ENDP

GDATA   LABEL BYTE                      ;パターン・データ
        DB      0,20H,2,0,40H,18H,4,0
        DB      0C8H,0C0H,18H,20H,40H,14H,1,090H
        DB      65H,48H,48H,30H,14H,090H,8,20H
```

```
        DB      0,40H,20H,2,80H,42H,56H,0
        DB      65H,6AH,0DBH,0AAH,62H,0DDH,66H,0DBH
        DB      0EDH,0EBH,0BAH,0AAH,0E9H,097H,4DH,0BBH

        DB      40H,0,80H,12H,10H,0F9H,1,54H
        DB      22H,0ABH,3,57H,41H,0ACH,4,0FAH
        DB      0,0,0,1FH,0,0FBH,41H,5
        DB      2,3,2,3,5,5,6,0FBH
        DB      0F4H,0C0H,0DAH,0,0B8H,0F8H,51H,4
        DB      0A2H,3,52H,3,61H,4,60H,0F8H

        DB      0,0,48H,4,09FH,11H,2AH,88H
        DB      0D5H,40H,0EAH,0C0H,35H,80H,5FH,20H
        DB      0,0,0F8H,0,0DFH,11H,0A0H,88H
        DB      0C0H,40H,0C0H,44H,0A0H,0A0H,0DFH,60H
        DB      1,72H,0,2DH,1FH,17H,20H,89H
        DB      0C0H,4AH,0C0H,44H,20H,85H,1FH,3

        DB      4,092H,4,09AH,9,1,8,0B0H
        DB      13H,49H,0,11H,2CH,0A0H,2,8
        DB      0FH,0FFH,1FH,0DFH,3FH,0C1H,39H,0F0H
        DB      35H,0FFH,82H,0BFH,80H,9,0,40H
        DB      0C0H,1,80H,0,0,1,0,0
        DB      0,0,80H,0,0C8H,2,0A9H,64H

        DB      49H,20H,59H,20H,80H,090H,9,8
        DB      091H,80H,2EH,20H,14H,0A8H,0,090H
        DB      0FFH,0F0H,0FBH,0F8H,83H,0FCH,0FH,09CH
        DB      0FFH,24H,0FAH,0B0H,36H,8,0,10H
        DB      80H,3,0,1,80H,0,0,0
        DB      0,8,0,0,0,0DH,42H,3BH

GDTOP   LABEL BYTE                              ;マップデータ
        DB      0,0,0,0,1,2,0,0,0,3,4,0,0,0,0,0,0,0,0,0,0,3,4
        DB      0,0,0,0,3,4,0,0,0,0,0,0,0,1,2,0,0,0,1,2,0,0,0,0
        DB      0,1,2,0,0,0,0,0,0,0,1,2,0,3,4,0,0,0,3,4,0,0,0,0
        DB      0,3,4,0,0,0,0,1,2,0,3,4,0,0,0,1,2,0,0,0,0,1,2,0
        DB      0,0,0,0,0,0,0,3,4,0,0,0,0,0,0,3,4,0,0,0,0,3,4,0
        DB      0,0,0,1,2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
        DB      0,0,0,3,4,0,0,0,0,0,0,1,2,0,0,0,0,0,0,0,0,0,0
        DB      0,0,0,0,0,1,2,0,0,0,0,3,4,0,1,2,0,0,0,0,1,2,0,0
        DB      1,2,0,0,0,3,4,0,0,0,0,0,0,0,3,4,0,0,0,0,3,4,0,0
        DB      3,4,0,0,0,0,0,0,0,0,1,2,0,0,0,0,0,0,0,0,0,0,1
        DB      0,0,0,0,0,0,0,0,0,0,3,4,0,0,0,0,0,1,2,0,0,0,0,3
        DB      0,0,1,2,0,0,0,0,1,2,0,0,0,0,0,0,0,3,4,0,0,0,0,0
        DB      0,0,3,4,0,0,0,0,3,4,0,0,0,0,0,0,0,0,0,0,0,1,2,0
        DB      0,0,0,0,0,0,0,0,0,0,0,0,0,1,2,0,0,0,0,0,0,3,4,0
        DB      0,0,0,0,1,2,0,0,0,0,0,0,3,4,0,0,0,1,2,0,0,0
        DB      2,0,0,0,3,4,0,0,0,1,2,0,0,0,0,1,2,0,3,4,0,0,0
        DB      4,0,0,0,0,0,0,0,0,3,4,0,1,2,0,0,3,4,0,0,0,0,0,1

TXCLR   LABEL BYTE                              ;テキスト・クリア　&　文字の属性の保存
        DB      1BH,"〔2J"
        DB      1BH,"〔s",1BH,"〔>5h",1BH,"〔1>h$"

TKEEP   LABEL BYTE                              ;文字の属性の復元
        DB      1BH,"〔u",1BH,"〔>5l",1BH,"〔1>l$"

KPMDT   LABEL BYTE                              ;画面保存用ワーク
        DW      GYMAX*16*GXMAX*3 DUP(?)

CODE    ENDS
```

```
STACK    SEGMENT STACK                             ;スタック・セグメントの定義
         DW      100H    DUP(0)
STACK    ENDS

BLUE     SEGMENT AT 0A800H                         ;B面のセグメントの定義
BLUE     ENDS

RED      SEGMENT AT 0B000H                         ;R面のセグメントの定義
RED      ENDS

GREEN    SEGMENT AT 0B800H                         ;G面のセグメントの定義
GREEN    ENDS

         END
```

芝生の中から無数のモグラが顔を出したような画面が表示されましたね。この時の表示は上から順にパターンを並べたものですが、見るからに平凡です。実は、これは平凡さを確認すると同時に、この画面からベタ・データを読んでいるのです。つまり、このデータをタイトルなどへの化粧表示へ利用しようというのです。化粧したほうの表示／消去はテンキーの1～3が対応しています。それぞれの内容は次のようなものです。なお、SHIFT キーを押している間はウェイトがかかるようになっていますから、実行過程をじっくりと見つめながら研究することもできます。

1：渦巻型の表示／消去（表示はマップパターンによる）

特にテクニックというものはありませんが、パターンを表示する順番におしゃれをしたものです。プログラムが面倒な割にはハデさはなく、おまけに表示に要する時間も結構長いため（ウェイトがないと渦巻に見えない）、何度も繰り返されるとカッタルイかもしれません。

2：ジワジワ表示／消去（表示はマップパターンによる）

画面を全体的にジワッと表示しジワッと消去するのは、ハイセンスなおしゃれといえます。おそらく画面を見ただけでは、表示／消去ともにアルゴリズムがわからないでしょう。実は、表示は各パターンを1バイト（B／R／G）ずつ計16回に分けて表示しているのです。それが全体的にジワッと見える要因なのですが、その1バイトを単に上から順番に表示したのでは、ラインごとに表示したようにしか見えません。そのため、2バイト×16ドットのパターンを図 3-4-2のような順で表示し、規則的にバラつかせているのです。

①	⑭
⑪	⑧
⑤	②
⑮	⑫
⑨	⑥
③	⑯
⑬	⑩
⑦	④

図 3-4-2　ジワジワ表示の順序

一方、消去のほうは1バイトを2ドットずつ計4回で消去していますが、これも1ラインおきに消去の向きを変えています。図 3-4-3を見て確認してください。正体がわかれば、どうということはないテクニックでしょう。

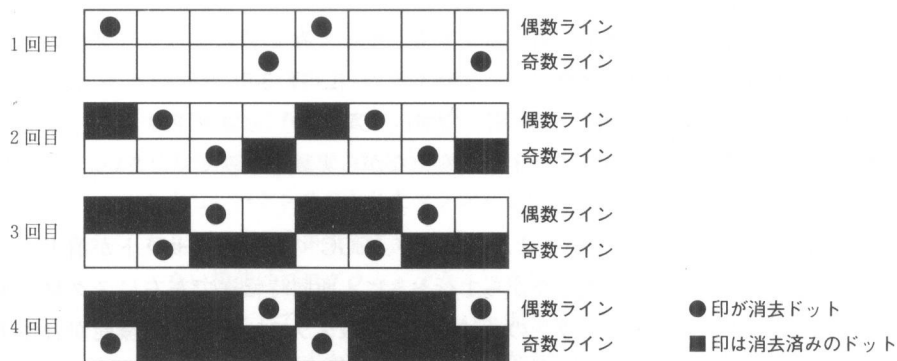



図 3-4-3　ジワジワ消去の順序

3：斜め表示／消去（表示はベタ・データによる）

考え方としては、これもジワッと表示させるための1つの手段です。プレーンごとに斜めに表示（消去）していき、画面エンドになったら再びその座標をゼロに戻してループさせ、表示（消去）が全体的に行われるようにするものです。これは、3面同時に表示（消去）するよりも、プレーンごとに実行したほうがキラキラしてきれいに見えます。ここでは、ループの手順として、X方向が右エンドになったらY座標は変化させずにX座標だけを0にしていますが、場合によってはY座標

①

1	8	15
10	2	9
4	11	3
13	5	12
7	14	6

OK

1	11	6
7	2	12
13	8	3
4	14	9
10	5	15

OK

②

1	6	11	16
13	2	7	12
9	14	3	8
5	10	15	4

OK

1			
	2		
		3	
			4

×（5番目が1番目と同じになってしまう）

③

1			4
	2		
		3	

×（5番目が1番目と同じになってしまう）

1	10	7	4
5	2	11	8
9	6	3	12

OK

図 3-4-4　斜め表示のアルゴリズムによる違い

も変化させなければならないことがあります。例えば、図 3-4-4の①ではどちらでもOKですが、②は前者の方法、③は後者の方法でないと同じ座標に戻ってしまいうまくいきません。

この区別はプログラムで判断するよりも、やってみてダメならプログラムを手直しするという方法が最も簡単です。プログラム（LIST 3-8）中のラベル（D2COK／C2COK）の上の行（JMP　CGTHD／JMP　CV2AD）をカットすれば、後者のアルゴリズムになりますので、サイズを変えながら実験してみてください。

今回は消去に EGCを用いませんでしたが、EGC のビットリセットが消去には最適です。ドット単位で消去しながら1行おきに4面同時転送で左右にスクロールアウトするとか、フェードアウトのイメージと拡張グラフィックスの機能が合体すると、より美しいおしゃれができるかもしれません。

また、おしゃれの陰に隠れて目立ちませんが、このプログラムにはアセンブラの基礎化粧ともいうべきテクニックが潜んでいます。それは、ラベルの用法についてです。表示位置（DVRAM）、画面サイズ（GXMAX／GYMAX）、マップサイズ（MXMAX／MYMAX）はラベルで定義されていますが、これに関係するプログラム中の数値もすべてこのラベルによる計算式となっています。そのため、表示位置や各サイズの変更は定義されたラベルの値を変更するだけで済み、プログラム本体はまったくいじる必要がありません。ゲーム制作途中で画面デザインを変えたい……というようなことでバグなど出さぬよう、こんな基礎化粧品があることも忘れてはなりません。

第4章　スクロール・テクニック

PC-9801 シリーズは、いわゆるゲーム専用機ではない本格パソコンとして商品化されています。サウンドはオプション（一部機種のみ標準装備）ですし、スプライト*のようなゲーム・マシンに必携の機能もありません。しかし、バランスのとれたマシンとして、ゲーム中心の若者の間に広く普及しているのも事実です。

ない機能はプログラム・テクニックでカバーする……当然のことですが、結果としてゲーム専用機のハードウェア機能を越えるテクニックを産み出したのですから、世の中なにが幸いするかわかりません。最近のゲームでは、多重重ね合わせ／多重スクロールといったゲーム専用機の上をいく技術も見られます（最新のゲーム専用機ではそういった機能も用意されていますが……）。

内部のプロセスはともかく、ゲーム専用機にヒケを取らないゲームが作られているという事実は、ある意味で98もゲーム専用機のジャンルに入るのかもしれません。

スプライト：

背景との重ね合わせや移動時の消去などを意識せずにキャラクタを表示できる特殊なグラフィックス機能のこと。

4－1　スクロールとキャラクタ

スクロールを辞書で調べると「巻物」という意味です。これが転じてスクロールゲームとなったわけですが、背景が一方的に動いていく初期の縦スクロールゲームは、まさに巻物というイメージにピッタリでした。

時は流れ、横スクロール、縦横スクロール、縦横斜めスクロール……と次第に巻物のイメージは薄れ、最近では多重スクロールという立体感あるものまで登場しています。これも技術の進歩があればこそなのですが、この技術を支えているのは EGCの拡張モード機能を含んだハードウェアの進化です。それだけに、対応機種がすべて EGC搭載機種に限られてくるのも仕方のないことでしょう。

ところで、スクロール・ゲームといえば忘れてならないのが『マシン語ゲームプログラミング』掲載の『スカイ・ブルーザー』です。88版／98版ともに、内容的には単純な縦スクロール・ゲームですが、テクニックはまったく別のものです。98版では GDCを利用したハードウェア・スクロールでしたが、88版は部分描き換えという旧式（？）の手法を使っています。とはいえ、これは今でも重要なスクロール技法の１つとなっており、特にハードウェア・スクロール機能のない PC-8801では、拡張グラフィックス（３面同時転送）による画面データ転送か、部分描き換えによる画面疑似移動によってのみスクロールは実現されるのです。

ただ、いずれの場合も問題は背景とキャラクタをどう区別するかです。88版『スカイ・ブルーザー』ではこれをプレーン分割によって解決していましたが、今ではフルカラーで重ね合わせ処理を行いながらスクロールさせるのは当然のこと。実際には、これこそがスクロール・テクニックの正体ともいえるくらいです。

さて、こういったテクニックが成立するためには、まずスクロールの条件を確定する必要があります。実際のゲームであれば、敵の出現、歩ける／歩けないの区別、入口やトラップの区別、高さ／奥行きの情報……等、ゲームに合わせてマップデータの構造を細かく設定することから始めなければなりません。とりあえず、ここではマップパターン／スクロールの単位を横16ドット×縦16ドットとすることだけを条件にテクニックに迫ってみたいと思います。

この単位は、もちろんゲームによって自由に設定して構いませんし、これがベストというわけでもありません。しかし、最近のゲームの傾向と多重重ね合わせ等の高度なテクニックには、この程度の単位がちょうどいいのです。その理由は、スクロールの滑らかさと速度のバランスが適度に同居し、マップパターンと同一サイズということでプログラミングがしやすいからです。

では、最初に88版『スカイ・ブルーザー』で使った部分描き換え方式をさらに発展させたEGCの拡張モード／４面同時転送機能によるスクロールを考えてみましょう。図 4-1-1 は最も基本的なスクロール画面で、画面中央にいるキャラクタが下方向に歩いたときの処理状態を示したものです。疑似３Ｄ処理のアクション RPGなどでよく見かける光景です。

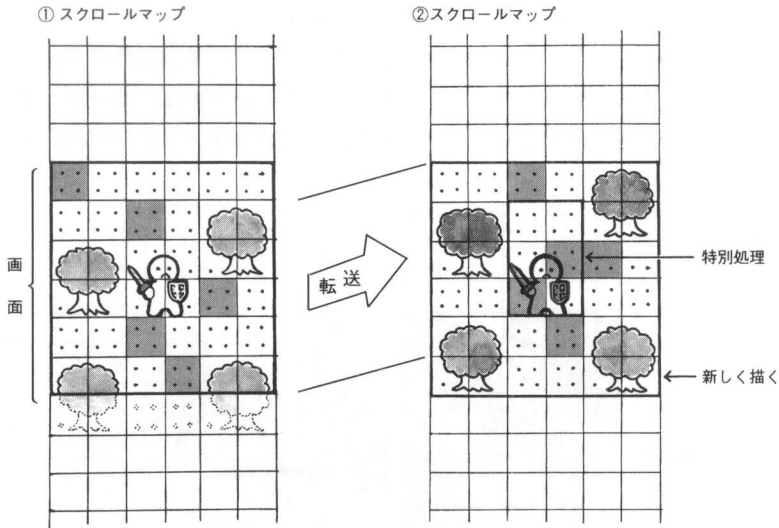



図 4-1-1　スクロールとキャラクタの関係

原理的には画面の２段目を１段目へというふうに転送し、最下段に新しく現れるマップパターンを描けばいいのですが、キャラクタが関わる特別処理枠内は①のデータをそのまま転送するわけにはいきません。これに対する最も単純な解決策は、キャラクタ部分を一旦背景で消去し、画面転送により新しい背景が完成してから新たなキャラクタを表示するという方法です。

しかし、この方法はキャラクタ不在の時間が長く、キャラクタがチラつくという致命的な欠点があります。そこで、最初に特別処理部分をグラフィックVRAMの余り（オフセットアドレス7D00H ～）に作成しておき、横１ラインを次のように分割して画面転送するのです。

特別処理枠の左側：通常の画面転送
特別処理枠の部分：7D00H 以降から転送
特別処理枠の右側：通常の画面転送

プログラム的にはレジスタのヤリクリが大変ですが、こうすることによりキャラクタの描き換え（消去／表示）とスクロールが同時に行われることになり、チラツキはまったくなくなります。もちろん、方向別に違ったプログラムを用意しなければなりませんが、基本的な考え方は同じです。画面転送プログラムを組む際には、方向（転送先）によってキチンと ディレクション・フラグを使い分けることも忘れないでください。

なお、画面がスクロールした場合、敵など主人公以外のキャラクタは相対的に反対方向へ移動したことになりますから、それらはすべてスクロールに合わせて座標を変化させなければなりません。これは『スカイ・ブルーザー』のように一方的な背景スクロールにはなかった意識の変化で、キー操作によってスクロールをコント

ロールする場合の基本的な注意事項です。

次に、もう1つのスクロール・テクニック、部分描き換えで図 4-1-1と同じ内容を実現してみましょう。部分描き換えというのは、元の絵と違う部分(パターン) だけを描き換えるということですから、図 4-1-2に示される部分を新しく表示し直すわけです。

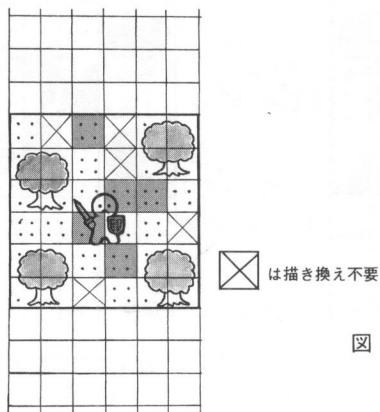



図 4-1-2　部分描き換えによるスクロール

この例では5カ所が描き換え不要となりますが、マップによってあるいはスクロールする方向によってもその数は変わってきます。全面描き換えでは滑らかなスクロールが得られませんから、マップ作成の時点でできるだけ同じパターンが連続するよう配慮をしなければなりません。つまり、マップにある程度の制限が加えられるわけで、この点に関する限りは自由にパターンを組み合わせられる画面転送方式のほうが優れていると言えるでしょう。

また、上下にスクロールするときはパターンを上から順に（横列ごとに）、左右にスクロールするときは左から順に（縦列ごとに）描き換えるという配慮もスクロール時のブレを解消するテクニックの1つです。もちろん、描き換えるパターン数が増えれば、ブレは消えても新たに波打ち現象が発生しますから、スクロール方向に合わせて同一パターンを連続するという努力は常に欠かすことができません。

こうして見ると、部分描き換え方式にはメリットがなさそうですが、実際にはこちらのほうが発展性のあるスクロールなのです。というのは、スクロール処理とキャラクタ表示をパターン単位で同時に行うため、複数マップによる多重スクロール(複数の背景＋キャラクタの合成）が可能になるからです。

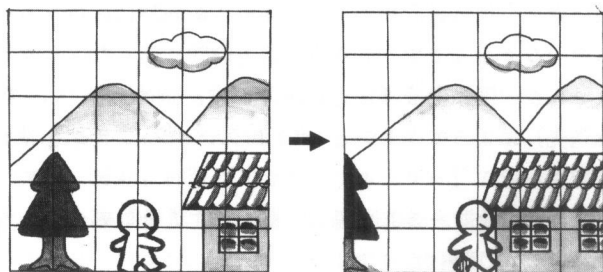



図 4-1-3　多重スクロール

例えば、図 4-1-3では山や雲の部分はそのままで、手前の木や家だけが横スクロールしていますが、画面転送ではこういった部分スクロールは不可能です。しかも、マップの枚数を追加してマップごとに移動頻度を変化させれば、さらに立体感のある画面を構成することができます（図 4-1-4)。

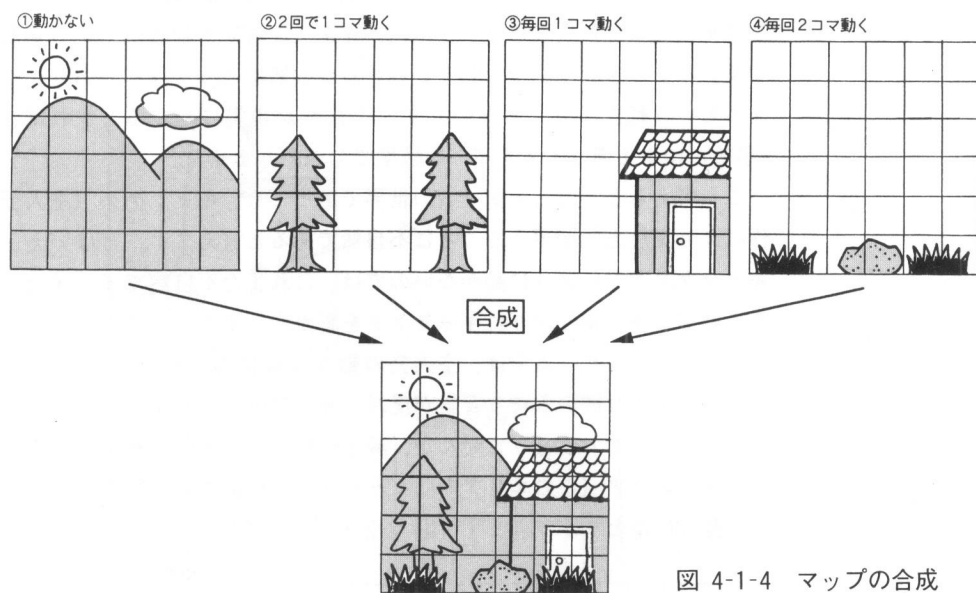



図 4-1-4　マップの合成

背景の優先順位は①→④の順に高くなり、それにつれて1回のスクロールでの動きも大きくなります。これが多重スクロールによる奥行きの表現です。この図ではキャラクタを表示していませんが、キャラクタの扱いは従来のように単体での消去／表示という考え方ではなく、どちらかというとスプライトに近くなります。つまり、キャラクタも複数のパターンによって構成されたマップの一種と考え、座標やパターンの変化を部分描き換えの対象とするわけです。

当然、キャラクタをどのマップ上に重ね合わせるかという情報も必要です。常識的には2番目くらいの優先順位（③の上に重ね合わせる）となるはずですが、場合によってはキャラクタ間で変化をつけるのも味なものです。ただし、キャラクタどうしで重なった場合は、キャラクタ番号で優先順位をつけるのが普通です。このあたり、いかにもスプライト的な処理であり、ソフト・スプライトと言ってもいいかもしれません。

この節では、具体的なプログラム例は示しませんが、こういった高度なテクニックはアルゴリズムを正確に把握することが実践への第一歩です。まずは、最近のゲームを思い出しながらスクロールの基礎原理を理解してください。

4－2　スクロールと非スクロール

未確認飛行物体……。いわゆるUFOは、慣性を無視したジグザグ飛行や急発進／急停止を平気で行うそうです。地球上の乗り物であれば、方向を変換するには放物線を描くように反転するか、減速→停止→逆方向への加速という過程を経るはずです。つまり、そういった動きがわれわれ人間にとって自然であり、いきなり反転するという動きは不自然なのです。不自然が現実に起こるとすれば、それは衝突（事故）による慣性エネルギーの破壊でしかありません。

こう考えると、たとえゲームの世界であっても、キャラクタ（主人公）がキー操作により突然方向変更することは不自然であると言えます。とはいえ、ゲーム中にキャラクタが思い通りに動かないのでは、これまた不自然でイライラしてしまいます。結局、キー操作通りにキャラクタを動かすしかないのですが……。

スクロール・ゲームでは、主人公の動きは背景のスクロールを意味しますから、方向転換をすればいきなり背景が反対方向へ動くことになります。これが絶対的に悪いということではありませんが、減速～停止～加速に代わる緩衝剤として非スクロール・エリア（このエリア内はキャラクタが移動する）を設けてやると、スクロールの動きにゆとりができます（図 4-2-1）。

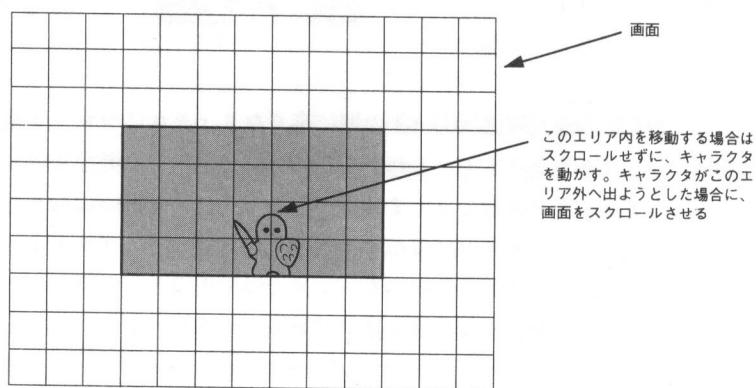



図 4-2-1　非スクロール・エリアの例

これは、要するにキャラクタ（主人公）が画面中央に常駐しているのか、ある程度動き回るのかという違いなのですが、これによってマップが広く感じられるという利点も生まれてきます。というのは、いずれにせよマップエンドになれば画面の端（通常は障害物でマップの周囲は閉鎖されている）までキャラクタが動くわけですが、その時に初めて画面中央から移動するよりも、普段から一定の範囲内を動いているほうがマップエンドの意識が少なくなるからです。ただし、非スクロール・エリアを広く取れば取るほど、画面の端から出現する敵との間合いが狭くなります。ゲームの内容と条件をよく考えて、そのことがネックにならないようにしなければなりません。

参考例として、図 4-2-2を見てください。これは断面図タイプのスクロールですが、AからBへキャラクタ（主人公）がジャンプする際の動き（①→③）を示したものです。非スクロール・エリアがあるせいで、頂点までスクロールアップした後はキャラクタ自身が落下（③）しており、スクロールはありません。たとえ右側のタナがない場合でも、スクロールダウンするのは非スクロール・エリアから出ようとする時点ですから、視覚的にも目が慣れて見やすくなります。

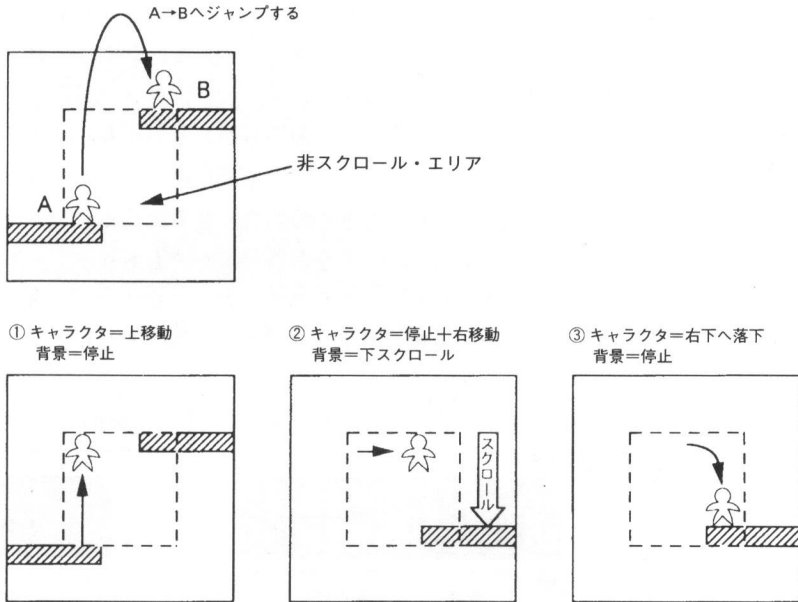



図 4-2-2　スクロールと非スクロール

こういった動きの基本にプログラム・テクニック（1節の内容）が加わって、初めてスクロール・テクニックとなるわけですが、これだけではまだプログラムを組むことはできません。スクロール・ゲームにとって背景をスクロールさせる（またはキャラクタを動かす）ことは、プログラム以前の必要最低条件に過ぎないのです。重要なのは、例えば図 4-2-2でいうならば、どうしてAからBへジャンプした際にタナから落下しないのかということなのです。

これまではマップを単なるパターン番号の集まりとして認識してきましたが、実際には1節でも触れたように、敵の出現、歩ける／歩けない、入口、各種トラップ、高さ（多重重ね合わせの場合）、奥行き（多重スクロールの場合）……等、ゲームに合わせて様々な情報が必要です。そして、これらの情報にこそ本当のテクニックが潜んでいるのです。

キャラクタや画面をコントロールするのはテクニックの氷山の一角、いわばスクロールの当然の知識です。水面下には画面に現れない多くの情報が隠されており、ここにゲームの実体があるのです。

4−3　マップの秘密

アクションタイプの RPGでは、あまり数字遊びという感覚はありませんが、思考型の場合は所持金やH.P.／M.P.といったものをヤリクリするのに相当頭を使います。ある意味では、それがゲームそのものであり、イベントや謎などは「数字遊び」をゲームとして成立させるための手段のような気さえします。

最近流行の戦略シミュレーション・ゲームも、シナリオはどうあれ結局は数字をもてあそぶゲームですし、「アチラを立てればコチラが立たず」という数字のお遊びは、それだけでパズルとしての面白さがあるのかもしれません。

実は、似たようなことはプログラムを組む前の段階から行われています。限られたメモリの中に、いかにして多くの情報を組み入れるか……。これは、プログラミング前の頭のトレーニング、すなわちパズルでもあります。例えば、『スカイ・ブルーザー』では背景パターンを0～127で表し、ビット７を敵出現のフラグにしていました。当然、敵の種類を示すマップが別に必要となります（図 4-3-1)。

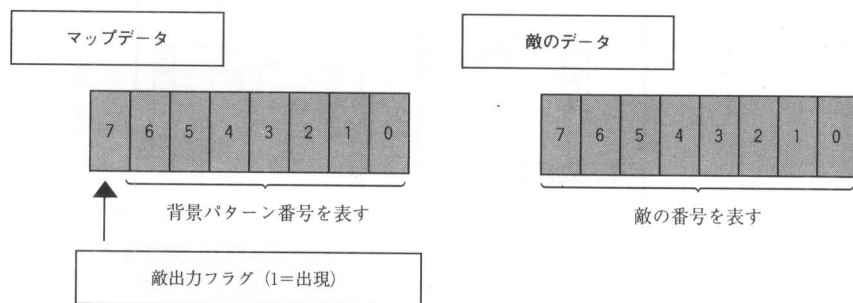



図 4-3-1　『スカイ・ブルーザー』のマップデータ構造

この場合、一方通行型のスクロールということで敵マップから不要部分（敵の出現しない部分）をカットしていましたが、自在にスクロール方向を選択できるタイプでは、敵マップもベタで持たないと、背景マップと一致させるのが大変です。その結果、マップに必要なメモリ数は単純に背景マップの２倍となります。この２枚のマップは、多重スクロール用のマップのように独立したものではなく、ビット不足を補うためのものですから、マップを２バイト単位（背景用１バイト＋敵用１バイト）とすれば、形の上では１つのマップにすることもできます。

このような方式で次々とその他の情報を追加していくと、マップデータはどんどん膨らんでしまい、結果としてオンメモリでのマップは小さくなる一方です。そこで、若干の制限はありますが、情報が１バイトに納まるようデータに工夫を凝らしてみましょう。

【制限事項】

敵の種類　　　　：16種（ 0～15）以下に限定

敵の出現する背景：パターン番号 0～7 に限定

この制限がゲームに及ぼす影響は、ほとんどゼロに近いものです。というのは、オンメモリでは敵の種類は16もあれば十分ですし、出現位置もこれだけの選択余地があればまず支障はないからです。それどころか、この程度の制限が問題になるようであれば、それはマッピングの努力不足というべきもの。マップ・エディタさえ制限に合わせて作成しておけば、実際には悩むこともないはずです。

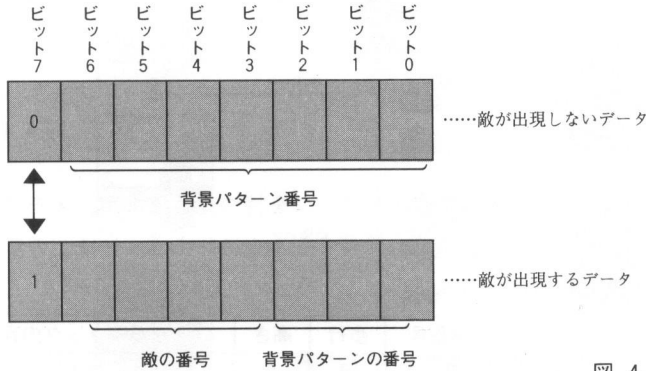



図 4-3-2

データの構造は、敵の出現フラグ（ビット7）が立っていなければ従来通り、立っている場合にはビット3～6で敵の番号を表すというものです。貴重なメモリですから、ゲームデザインからの協力も必要なのです。

一方、歩ける／歩けない、入口やトラップ……など、場所に関する情報を追加したい場合は、パターン番号で区別します。当然、プログラム側で区別しやすいようパターンを種類別にまとめておくことが大切です。ただし、特定の入口や個々に内容が違うトラップの場合は、さらに座標で区別することになります。

また、ふかん図のような疑似3Ｄ画面では、よく逆重ね合わせのテクニックが使われます。例えば、人が柵の手前にいる時は人を優先し、柵の奥にいる時は柵を優先して表示するというぐあいです（図 4-3-3）。

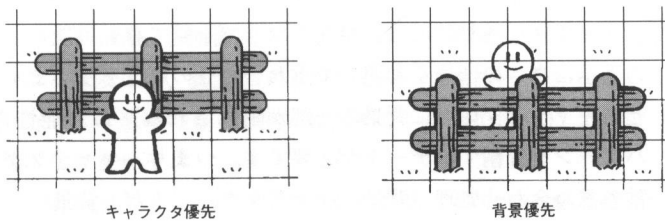



図 4-3-3　逆重ね合わせ

こういったことを実現するには、背景パターン／キャラクタに高さの情報を持たせ、それぞれの高さを比較して優先順位をつければいいのです（図 4-3-4）。

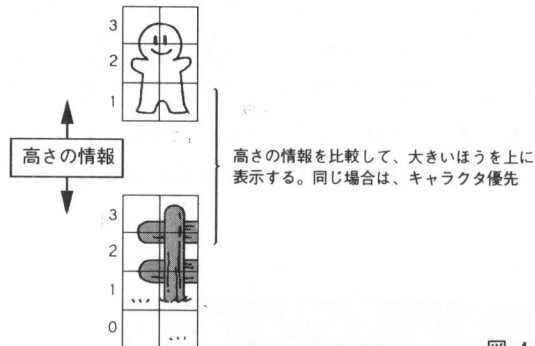



図 4-3-4　高さの情報

これも、理想は高さ専用のマップを持つことですが、メモリ節約のため同様にパターン番号で高さの情報を管理するようにします。先ほどの歩ける／歩けないの情報と試しに組み合わせてみましょう。

パターン番号	歩行	高さ	グラフィック内容
0〜19	可	0	地上
20〜39	不可	1	棚の下部や木の根元など
40〜49	可	2	棚の中間部や木の幹など
50〜69	可	3	棚の上部や木の上部など
70〜79	不可	2	家の壁や屋根などの一部
80〜89	可	0	トラップ（落とし穴）など
⋮	⋮	⋮	⋮

地上のパターン（0〜7）には敵出現の情報が入りますから、同系統のパターンが片寄らないようにする工夫も必要です。また、地形からどうしてもそれが無理な場合は、特殊処理としてパターン番号で敵を出現させることもあります。

ここに示した内容は単なるサンプルに過ぎませんが、このようにパターン番号に情報を定義すると、ほとんどの情報は1枚のマップに収めることができ、オンメモリで広い世界を表現することができるようになります。

キャラクタ移動時には、足元のマップを見て移動可能かどうかを判定し、移動後の表示は高さの情報を参照して重ね合わせの順位を決めます。この時、逆重ね合わせ（キャラクタの上に背景の一部が表示される）の可能性のある背景パターンは、パターンを分割するデータが必要です。つまり、キャラクタが上位にくる場合は通常の重ね合わせ処理（背景＋キャラクタ）ですが、背景の一部が上位にくる場合は、背景パターンを地面（キャラクタより下位に表示される）と柵（キャラクタより上位に表示される）とに分ける必要があるということです。

これをプログラムで実現すると、次のような処理手順となります。なお、ここでの手順は EGCの機能を有する機種を対象としており、EGCの拡張モード機能を利用することを前提としています。

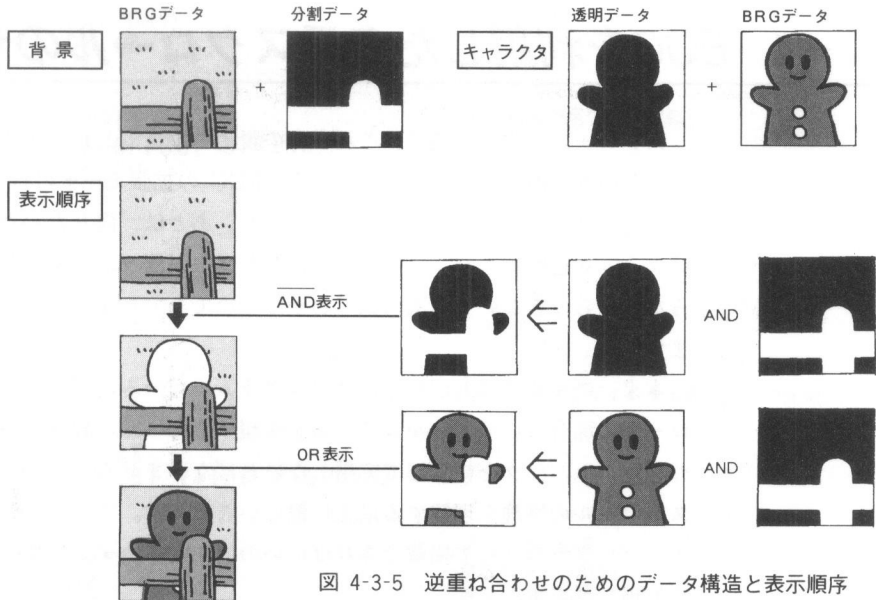



図 4-3-5 逆重ね合わせのためのデータ構造と表示順序

①背景パターンを表示
②透明データで分割画面をくり抜く
③アクティブプレーンをB面のみとして、B面用データをG.VRAMへ格納する
④アクティブプレーンをR面のみとして、R面用データをG.VRAMへ格納する
⑤アクティブプレーンをG面のみとして、G面用データをG.VRAMへ格納する
(注) I面のデータがある場合にはI面に関しても、同様の処理をする

この一連の処理は、グラフィックVRAMの余り (7D00H以降) で行ってから実際の表示先へ転送したほうがいいことは、すでに第2章『2—5 グラフィックVRAMの余り』で述べたとおりです。スクロールというのは、最も速度を要求されるテクニックですから、すべての面で高速化を追求しなければならないのです。

なお、歩ける/歩けないのチェックについては、方向によって条件が変化する場合もあります。特に、断面図タイプのスクロールでは、図 4-2-2のようにタナがズレているとは限らず、真上にタナがあるケースも少なくありません。このような場合、上方向には進めても下方向には進めない (落ちない) ようにプログラムを組まなければなりません。

また、断面図タイプで上方向へ進む時には、足元ではなく頭の部分で進める/進めないのチェックをします。その際、見かけは同じようでも天井の場合もありますから、両者はキチンと区別をつけておかなければなりません。いずれにしても、この段階でパターン番号に意味を持たせて整理するということは、プログラム前のパズルであり、最初の難所といっても過言ではないでしょう。

4－4　EGCを利用した多重スクロールのサンプル

スクロールに関する限り、これまでの説明で少なくとも「わからないからできない」という段階は過ぎたことでしょう。EGC の拡張モードの機能も知っているし、重ね合わせの方法もマスターしてきました。あとは、作りたいゲームに合わせてプログラムを組むばかり。……と、すんなりコトが運べば苦労はしませんね。「わからないからできない」の次には「わかっていてもできない」という難関が控えています。

本来、アルゴリズムがわかればプログラムは組めるはずですが、フルカラースクロールの場合は、そのアルゴリズムを実現するためにこれまでに紹介してきたいくつものテクニックを複合して応用しなければなりません。実際問題として、これはスクロールの原理を理解する以上に難しいことです。すべてのスクロールをサンプル・プログラムとして掲載できればいいのですが、ページに限りのある本ではそれも無理な話です。

そこで、数あるスクロールの中から最も新しいテクニックである多重スクロールを取り上げ、これを徹底的に解析してこの難関を乗り越えてもらうことにしました。多重スクロールというのは、部分描き換えによる単純スクロールを発展させたものですが、断面図タイプの画面に奥行き感を出したり、立体駐車場のような階層を表現できるので、今後の応用が期待されているテクニックのひとつです。

また、スクロール処理と切り離せないのがキャラクタとの合成です。『**4－1　スクロールとキャラクタ**』でソフト・スプライトという言葉が出てきましたが、キャラクタをスクロール画面と無関係に動かすことは、プログラム的にはスクロール以上に面倒なことです。しかし、ソフト・スプライトを省いてしまってはゲームになりませんね。基本的な考え方は『**4－1　スクロールとキャラクタ**』に示したとおりですが、実際には複雑な処理を経て重ね合わせ処理が実現されています。

では、まずはサンプル・プログラムを実行して、こういった２つのテクニックを自分の目で確認してください。

```
A>LIST4-1 ⏎
```

LIST 4-1

```
;*************************************
;*              LIST4-1              *
;*************************************

EXTRN   GSTAT:NEAR,GMD16:NEAR,GTERM:NEAR,EGC_ON:NEAR,EGC_OF:NEAR

CODE    SEGMENT PUBLIC

        ASSUME  CS:CODE,DS:CODE

OUTPORT MACRO   PORT,DATA                              ;;ポート出力用マクロ定義
        MOV     AX,DATA
```

```
        MOV     DX,PORT
        OUT     DX,AX
        ENDM

PRINT   MACRO   STRING                          ;;文字列出力用マクロ定義
        MOV     DX,OFFSET STRING
        MOV     AH,09
        INT     21H
        ENDM

PUTPM   MACRO                                   ;;G.VRAMへのデータ格納用マクロ定義
        MOVSW
        ADD     DI,CX
        ENDM

TXYSET  MACRO   REG,T_X,T_Y                     ;;テキスト文字ダイレクト表示マクロ定義
        MOV     AX,TEXT
        MOV     ES,AX
        MOV     REG,&T_Y*160 + &T_X*2
        ENDM

DITOP   EQU     80*16+8                         ;マップ表示位置
DMAXX   EQU     32                              ;表示パターン数(X)
DMAXY   EQU     16                              ;表示パターン数(Y)

FPAT1   EQU     2                               ;この番号以降ベタ・パターン（背景②)
FPAT2   EQU     2                               ;この番号以降ベタ・パターン（背景③)

SPVAL   EQU     20                              ;総スプライト数
SPGVA   EQU     7D00H-SPVAL*32                  ;ソフト・スプライト用G.VRAM
URAGV   EQU     7D00H                           ;パターン合成エリア（G.VRAM)
GPT00   EQU     URAGV+32                        ;ベタ・パターンデータ（背景①／番号0)
GPT01   EQU     GPT00+32                        ;ベタ・パターンデータ（背景①／番号1)
GPT11   EQU     GPT01+32                        ;ベタ・パターンデータ（背景②／番号2)
GPT21   EQU     GPT11+32                        ;ベタ・パターンデータ（背景③／番号2)
GPT22   EQU     GPT21+32                        ;ベタ・パターンデータ（背景③／番号3)

PMAIN:  CALL    GSTAT                           ;グラフィックシステム・スタート
        JNB     $+5
        JMP     TEXIT                           ;異常終了であればTEXITへ
        CALL    GMD16                           ;640×400ドット4096色モード設定
        JNB     $+5                             ;
        JMP     TEXIT                           ;異常終了であればTEXITへ
        IN      AL,71H                          ;AL←現在のクロックの秒数
        MOV     RNDDT,AL                        ;乱数のワーク・エリア初期化
        PRINT   TXCLR                           ;テキスト画面クリア
        CALL    STINT                           ;割り込み処理の初期設定
        MOV     AX,BLUE
        MOV     ES,AX                           ;ES=BLUE
        CALL    EGC_ON                          ;EGCの拡張モードオン
        OUTPORT 4A0H,0FFF0H                     ;
        OUTPORT 4A2H,0FFH                       ;
        OUTPORT 4A4H,0CF0H                      ;  EGCの初期設定
        OUTPORT 4A8H,0FFFFH                     ;
        OUTPORT 4ACH,0                          ;
        OUTPORT 4AEH,<16-1>                     ;
        CLD                                     ;
        MOV     CX,4000H                        ;
        XOR     AX,AX                           ;  グラフィックス画面のクリア
        MOV     DI,AX                           ;
        REP     STOSW                           ;
        PUSH    ES                              ;
        TXYSET  DI,0,24                         ;
```

```
        MOV      CX,50H                       ;} テキスト画面を部分マスク
        MOV      AX,87H                       ;
TXLP1:  STOSW                                 ;
        MOV      BYTE PTR ES:[DI+1FFEH],1     ;
        LOOP     TXLP1                        ;
        POP      ES                           ;
        OUTPORT  4A0H,0FFFBH                  ;
        MOV      AL,0AAH                      ;
        XOR      DX,DX                        ;} 背景の描写
        CALL     BACKD                        ;
        OUTPORT  4A0H,0FFF1H                  ;
        MOV      AL,0AAH                      ;
        MOV      DX,1                         ;
        CALL     BACKD                        ;
        OUTPORT  4A0H,0FFFEH                  ;
        MOV      SI,OFFSET BKD00              ;
        MOV      DI,GPT00                     ;
        MOV      CX,32                        ;
        REP      MOVSW                        ;} データをG.VRAMへ転送
        CALL     GVTEN                        ;
        MOV      DI,GPT21                     ;
        CALL     GVTEN                        ;
        MOV      DI,GPT22                     ;
        CALL     GVTEN                        ;
        CALL     ALMAP                        ;初期画面用描き換え情報マップ作成
        CALL     DIMAP                        ;初期画面表示
MLOOP:  CALL     MVMP1                        ;
        CALL     MVMP2                        ;
        CALL     QSSET                        ;} 背景②③とピーヨを動かして表示
        CALL     PIYOM                        ;
        CALL     MAKSG                        ;
        CALL     DIMAP                        ;
        CALL     VWCHK                        ;ウェイトのチェック
        CALL     NOMP1                        ;
        CALL     MVMP2                        ;
        CALL     QSSET                        ;} 背景③とピーヨを動かして表示
        CALL     PIYOM                        ;
        CALL     MAKSG                        ;
        CALL     DIMAP                        ;
        CALL     VWCHK                        ;ウェイトのチェック
        MOV      AX,400H                      ;
        INT      18H                          ;
        TEST     AH,1                         ;} [ESC] が押されていなければMLOOPへ
        JNE      EGCOF                        ;
        JMP      MLOOP                        ;
EGCOF:  CALL     EGC_OF                       ;EGCの拡張モードオフ
        CALL     GTERM                        ;グラフィックシステムの終了
        CALL     ORIINT                       ;割り込み関係を元に戻す
        PRINT    TKEEP                        ;テキスト文字の属性を元に戻す
TEXIT:  MOV      AX,0C00H                     ;} キーボード・バッファ・クリア
        INT      21H                          ;
        MOV      AX,4C00H                     ;} MS-DOSシステムへ戻る
        INT      21H                          ;

VSYNC   PROC     FAR
        PUSH     AX                           ;
        INC      BYTE PTR CS:VWDAT            ;
        MOV      AL,20H                       ;
        OUT      0,AL                         ;} 割り込み処理ルーチン
        OUT      64H,AL                       ;
        POP      AX                           ;
        IRET                                  ;
VSYNC   ENDP
```

```
VWCHK   PROC
VWKIN:  MOV     AH,2                       ;
        INT     18H                        ;
        TEST    AL,1                       ;
        JNE     VWKIN                      ;
VWALP:  MOV     AL,VWDAT                   ; ウェイト（シフトが押されている間はループ）
        CMP     AL,5                       ;
        JB      VWALP                      ;
        XOR     AL,AL                      ;
        MOV     VWDAT,AL                   ;
        RET                                ;
VWCHK   ENDP
VWDAT   DB      0                          ;ウェイト用カウンター
;
IREDVSY EQU     350AH                      ;V-SYNC割り込みベクタを求めるため
;
ISETVSY EQU     250AH                      ;V-SYNC割り込みベクタをセットするため
;
KPMASK  DB      0
;
KPVNOFF DW      0                          ;KeeP VsyNc OFFset
;
KPVNSEG DW      0                          ;KeeP VsyNc SEGment
;
STINT   PROC                               ;
        MOV     AX,IREDVSY                 ;
        INT     21H                        ;
        MOV     KPVNOFF,BX                 ;
        MOV     KPVNSEG,ES                 ;
        MOV     AX,ISETVSY                 ;
        MOV     DX,OFFSET VSYNC            ;
        INT     21H                        ; 割り込み処理の初期設定
        CLI                                ;
        IN      AL,2                       ;
        MOV     KPMASK,AL                  ;
        AND     AL,0FBH                    ;
        OUT     2,AL                       ;
        OUT     64H,AL                     ;
        STI                                ;
        RET                                ;
STINT   ENDP
;
ORIINT  PROC
        PUSH    DS                         ;
        LDS     DX,dword ptr CS:KPVNOFF    ;
        MOV     AX,ISETVSY                 ;
        INT     21H                        ;
        CLI                                ; 割り込み関係を復元
        MOV     AL,CS:KPMASK               ;
        OUT     2,AL                       ;
        STI                                ;
        POP     DS                         ;
        RET                                ;
ORIINT  ENDP
;
BACKD   PROC
        MOV     AH,400/8                   ;
BACL0:  MOV     CL,40                      ;
BACL1:  MOV     CH,8                       ;
        MOV     DI,DX                      ;
BACL2:  STOSB                              ;
        ADD     DI,79                      ;
        DEC     CH                         ; 背景の描写
```

```
        JNE     BACL2                        ;
        ADD     DX,2                         ;
        DEC     CL                           ;
        JNE     BACL1                        ;
        ADD     DX,80*7                      ;
        XOR     DX,1                         ;
        DEC     AH                           ;
        JNE     BACL0                        ;
        RET                                  ;
BACKD   ENDP

GVTEN   PROC
        MOV     AX,0FFFEH                    ;
        MOV     DX,04A0H                     ;
        CALL    GTENS                        ;
        CALL    GTENS                        ;
        CALL    GTENS                        ;
        CALL    GTENS                        ;
        RET                                  ;
                                             ; データをG.VRAMへ転送
GTENS   PROC                                 ;
        PUSH    DI                           ;
        OUT     DX,AX                        ;
        MOV     CX,16                        ;
        REP     MOVSW                        ;
        POP     DI                           ;
        ROL     AX,1                         ;
        RET                                  ;
GTENS   ENDP
GVTEN   ENDP

MA1TB   DW      0,BKT11                      ;背景②重ね合わせパターン・テーブル
MA2TB   DW      0,BKT21                      ;背景③重ね合わせパターン・テーブル

DIMAP   PROC
        MOV     DI,OFFSET CKMAP              ;DI=描き換え情報マップアドレス
        MOV     DM0DT,OFFSET MAP00           ;
        MOV     DM1DT,OFFSET MAP01           ; 各マップアドレス初期化
        MOV     DM2DT,OFFSET MAP02           ;
        MOV     BX,SPGVA                     ;
        MOV     SPRDT,BX                     ;
        MOV     BP,DITOP                     ;BP=表示アドレス
        MOV     CL,DMAXX                     ;CL=表示パターン数(X)
DMPL0:  MOV     CH,DMAXY                     ;CH=表示パターン数(Y)
        PUSH    BP                           ;
DMPL1:  PUSH    CX                           ;
        MOV     AL,[DI+0]                    ;
        ROR     AL,1                         ;
        JB      $+5                          ;
        JMP     MPSAM                        ;
        ROR     AL,1                         ;
        JNB     $+5                          ;
        JMP     SPRIT                        ;
        ROR     AL,1                         ; 描き換え情報マップの内容により分岐
        JNB     $+5                          ;
        JMP     FULL2                        ;
        ROR     AL,1                         ;
        JB      CMAP1                        ;
        ROR     AL,1                         ;
        JNB     $+5                          ;
        JMP     FULL1                        ;
        ROR     AL,1                         ;
        JB      $+5                          ;
```

```
        JMP     FULL0                       ;
        CALL    DMAP0                       ;
        CALL    DMAP1                       ;
        MOV     BX,URAGV                    ; 背景①（フル表示）
        CALL    PUTPT                       ;   ＋背景②（重ね合わせ表示）
        JMP     MPSA1                       ;   ＋背景③（表示不要）

CMAP1:  ROR     AL,1                        ; 情報により分岐
        JNB     CMAP2                       ;
        MOV     BX,DM1DT                    ;
        MOV     AL,[BX]                     ;
        INC     BX                          ;
        MOV     DM1DT,BX                    ;
        SUB     AL,FPAT1                    ;
        MOV     AH,0                        ;
        SHL     AX,1                        ;
        SHL     AX,1                        ; 背景①（表示不要）
        SHL     AX,1                        ;   ＋ 背景②（フル表示）
        SHL     AX,1                        ;   ＋ 背景③（重ね合わせ表示）
        SHL     AX,1                        ;
        MOV     BX,GPT11                    ;
        CALL    DMASB                       ;
        CALL    DMAP2                       ;
        MOV     BX,URAGV                    ;
        CALL    PUTPT                       ;
        INC     WORD PTR DM0DT              ;
        JMP     CMAP4                       ;

CMAP2:  ROR     AL,1                        ; 情報により分岐
        JNB     CMAP3                       ;
        CALL    DMAP0                       ;
        CALL    DMAP1                       ; 背景①（フル表示）
        CALL    DMAP2                       ;   ＋背景②（重ね合わせ表示）
        MOV     BX,URAGV                    ;   ＋背景③（重ね合わせ表示）
        CALL    PUTPT                       ;
        JMP     CMAP4                       ;

CMAP3:  CALL    DMAP0                       ;
        CALL    DMAP2                       ; 背景①（フル表示）
        MOV     BX,URAGV                    ;   ＋背景②（表示不要）
        CALL    PUTPT                       ;   ＋背景③（重ね合わせ表示）
        INC     WORD PTR DM1DT              ;
        JMP     CMAP4                       ;

SPRIT:  MOV     BX,SPRDT                    ;
        CALL    PUTPT                       ;
        INC     DX                          ; スプライト・パターンの転送
        INC     DX                          ;
        MOV     SPRDT,DX                    ;
        JMP     MPSAM                       ;

FULL2:  MOV     BX,DM2DT                    ;
        MOV     AL,[BX]                     ;
        SUB     AL,FPAT2                    ; 背景③だけのパターン
        MOV     BX,GPT21                    ;
        JMP     FUL12                       ;

FULL1:  MOV     BX,DM1DT                    ;
        MOV     AL,[BX]                     ;
        SUB     AL,FPAT1                    ; 背景②だけのパターン
        MOV     BX,GPT11                    ;
        JMP     FUL12                       ;
```

```
FULL0:  MOV     BX,DM0DT                  ;
        MOV     AL,[BX]                   ;
        MOV     BX,GPT00                  ;
FUL12:  MOV     AH,0                      ;
        SHL     AX,1                      ;
        SHL     AX,1                      ; 背景①だけのパターン
        SHL     AX,1                      ;
        SHL     AX,1                      ;
        SHL     AX,1                      ;
        ADD     BX,AX                     ;
        CALL    PUTPT                     ;

MPSAM:  INC     WORD PTR DM0DT            ;
        INC     WORD PTR DM1DT            ; マップアドレスを変更
MPSA1:  INC     WORD PTR DM2DT            ;

CMAP4:  INC     DI                        ;
        POP     CX                        ;
        ADD     BP,80*16                  ;
        DEC     CH                        ;
        JE      $+5                       ;
        JMP     DMPL1                     ; 各レジスタを次パターン用に変更し、必要回数
        POP     BP                        ; だけループ
        INC     BP                        ;
        INC     BP                        ;
        DEC     CL                        ;
        JE      $+5                       ;
        JMP     DMPL0                     ;
        RET                               ;

SPRDT   DW      SPGVA
DM0DT   DW      OFFSET MAP00

DMAP0   PROC                              ;
        MOV     BX,DM0DT                  ;
        MOV     AL,[BX]                   ;
        INC     BX                        ;
        MOV     DM0DT,BX                  ;
        MOV     AH,0                      ;
        SHL     AX,1                      ;
        SHL     AX,1                      ;
        SHL     AX,1                      ;
        SHL     AX,1                      ;
        SHL     AX,1                      ;
        MOV     BX,GPT00                  ; 背景①の表示
DMASB:  ADD     BX,AX                     ;
        PUSH    SI                        ;
        PUSH    DS                        ;
        MOV     AX,BLUE                   ;
        MOV     DS,AX                     ;
        MOV     SI,BX                     ;
        OUTPORT 4A4H,28F0H                ;
        MOV     AX,0FFF0H                 ;
        CALL    DIURA                     ;
        POP     DS                        ;
        POP     SI                        ;
        RET                               ;
DMAP0   ENDP

DM1DT   DW      OFFSET MAP01
DM2DT   DW      OFFSET MAP02

DMAP1   PROC
```

```
        MOV      BX,DM1DT                          ;
        MOV      AL,[BX]                           ;
        INC      BX                                ;
        MOV      DM1DT,BX                          ; 背景②の表示
        MOV      BX,OFFSET MA1TB                   ;
        CALL     DICHR                             ;
        RET                                        ;
DMAP1   ENDP

DMAP2   PROC                                       ;
        MOV      BX,DM2DT                          ;
        MOV      AL,[BX]                           ;
        INC      BX                                ; 背景③の表示
        MOV      DM2DT,BX                          ;
        MOV      BX,OFFSET MA2TB                   ;
        CALL     DICHR                             ;
        RET                                        ;
DMAP2   ENDP

DICHR   PROC                                       ;
        ADD      AL,AL                             ;
        MOV      DL,AL                             ;
        MOV      DH,0                              ;
        ADD      BX,DX                             ;
        PUSH     SI                                ;
        MOV      SI,[BX]                           ;
        OUTPORT  4A4H,0C0CH                        ; マップパターンを7D00H(G.VRAM)から作成
        MOV      AX,0FFF0H                         ;
        CALL     DIURA                             ;
        OUTPORT  4A4H,0CFCH                        ;
        MOV      AX,0FFFEH                         ;
        CALL     DIURA                             ;
        CALL     DIURA                             ;
        CALL     DIURA                             ;
        CALL     DIURA                             ;
        POP      SI                                ;
        RET                                        ;
DICHR   ENDP                                       ;
                                                   ;
DIURA   PROC                                       ;
        MOV      DX,4A0H                           ;
        OUT      DX,AX                             ;
        XCHG     DI,DX                             ;
        MOV      DI,URAGV                          ;
        MOV      CX,16                             ;
        REP      MOVSW                             ;
        ROL      AX,1                              ;
        XCHG     DI,DX                             ;
        RET                                        ;
DIURA   ENDP

PUTPT   PROC                                       ;
        OUTPORT  4A4H,28F0H                        ;
        OUTPORT  4A0H,0FFF0H                       ;
        PUSH     DS                                ;
        MOV      AX,BLUE                           ;
        MOV      DS,AX                             ;
        MOV      CX,80-2                           ;
        MOV      DX,BX                             ;
        XCHG     SI,DX                             ;
        XCHG     DI,BX                             ;
        MOV      DI,BP                             ;
        PUTPM
```

```
        PUTPM                                ;} 作成済みのパターンを指定される位置に転送
        PUTPM                                ;
        PUTPM                                ;
        PUTPM                                ;
        PUTPM                                ;
        PUTPM                                ;
        PUTPM                                ;
        PUTPM                                ;
        PUTPM                                ;
        PUTPM                                ;
        PUTPM                                ;
        PUTPM                                ;
        PUTPM                                ;
        PUTPM                                ;
        MOV     AX,[SI]                      ;
        MOV     [DI],AX                      ;
        XCHG    SI,DX                        ;
        XCHG    DI,BX                        ;
        POP     DS                           ;
        RET                                  ;
PUTPT   ENDP
DIMAP   ENDP

MAKSG   PROC
        CALL    MLANK                        ;} 順位通りにスプライト表示パターンを作成する
        MOV     BX,SPGVA                     ;
        MOV     SGADT,BX                     ;
        MOV     AL,1                         ;
MSPLP:  MOV     BYTE PTR [MSPDT],1           ;
        MOV     BX,OFFSET SLANK+SPVAL-1      ;
        MOV     CH,SPVAL                     ;
MSPL1:  CMP     AL,[BX]                      ;
        JE      MSPG1                        ;
        DEC     BX                           ;
        DEC     CH                           ;
        JNE     MSPL1                        ;
        JMP     MSPRT                        ;

MSPG1:  PUSH    AX                           ;} 1ケ目のスプライト表示パターンを作成
        PUSH    BX                           ;
        PUSH    CX                           ;
        CALL    MGVDT                        ;
        POP     CX                           ;
        POP     BX                           ;
        POP     AX                           ;
        JMP     MSPG3                        ;

MSPG2:  DEC     BX                           ;} 同一順位のスプライトがあればパターンを作成
        CMP     AL,[BX]                      ;
        JNE     MSPG3                        ;
        PUSH    AX                           ;
        PUSH    BX                           ;
        PUSH    CX                           ;
        CALL    MGV1W                        ;
        POP     CX                           ;
        POP     BX                           ;
        POP     AX                           ;

MSPG3:  DEC     CH                           ;} 背景③のパターンを重ね合わせ、スプライト番号
        JNE     MSPG2                        ;  表示アドレスを更新する
        CALL    MGVD2                        ;
        ADD     AL,MSPDT                     ;
        ADD     WORD PTR SGADT,32            ;
        JMP     MSPLP                        ;
```

```
MSPRT:  RET                                     ;J

MLANK   PROC
        MOV     BX,OFFSET SPWOK                 ;
        MOV     DI,OFFSET SLANK                 ;
        MOV     CX,SPVAL                        ;
MLANL:  MOV     AL,[BX]                         ;
        OR      AL,AL                           ;  各スプライトに表示順位をつける
        JE      $+5                             ;
        CALL    CHLCK                           ;
        MOV     [DI],AL                         ;
        ADD     BX,3                            ;
        INC     DI                              ;
        LOOP    MLANL                           ;
        RET
MLANK   ENDP

CHLCK   PROC                                    ;
        PUSH    BX                              ;
        PUSH    CX                              ;
        INC     BX                              ;
        MOV     DH,[BX]                         ;
        INC     BX                              ;
        MOV     DL,[BX]                         ;
        MOV     BX,OFFSET SPWOK                 ;
        MOV     CX,SPVAL                        ;
        MOV     AL,1                            ;
CHLCL:  TEST    BYTE PTR [BX],01H               ;
        JE      CHLC2                           ;
        MOV     AH,DH                           ;  スプライトの表示順位を各座標を調べて求める
        SUB     AH,[BX+1]                       ;
        JB      CHLC2                           ;
        JNE     CHLC1                           ;
        MOV     AH,DL                           ;
        SUB     AH,[BX+2]                       ;
        JBE     CHLC2                           ;
CHLC1:  INC     AL                              ;
CHLC2:  ADD     BX,3                            ;
        LOOP    CHLCL                           ;
        POP     CX                              ;
        POP     BX                              ;
        RET                                     ;
CHLCK   ENDP

MGVDT   PROC
        MOV     AL,CH                           ;
        ROL     AL,1                            ;  スプライト・パターンのアドレスを求めるため
        MOV     BYTE PTR MGPDT,AL               ;
        ADD     AL,CH                           ;
        MOV     CL,AL                           ;
        MOV     CH,0                            ;
        MOV     BX,OFFSET SPWOK                 ;
        ADD     BX,CX                           ;
        DEC     BX                              ;
        MOV     CL,[BX]                         ;  背景③のマップアドレスを求めるため
        DEC     BX                              ;
        MOV     BL,[BX]                         ;
        MOV     BH,0                            ;
        SHL     BX,1                            ;
        SHL     BX,1                            ;
        SHL     BX,1                            ;
        SHL     BX,1                            ;
        ADD     BX,CX                           ;
```

```
        MOV     MP2DT,BX                ;J
        MOV     CH,BH                   ;]
        MOV     CL,BL                   ;|
        MOV     DX,OFFSET CKMAP         ;|
        ADD     BX,DX                   ;} 描き換え情報マップのアドレスをセット
        MOV     MG1DT,BX                ;|
        MOV     MG2DT,BX                ;J
        OR      BYTE PTR [BX],01H       ;情報マップに描き換え有りのフラグを立てる
        OR      BYTE PTR [BX],02H       ;情報マップにスプライト有りのフラグを立てる
        TEST    BYTE PTR [BX],04H       ;} 背景③がベタパターンならリターン
        JNE     MGVRT                   ;J
        TEST    BYTE PTR [BX],10H       ;]
        JNE     $+5                     ;} 背景②がベタパターンでないなら背景①を作成
        CALL    SFMP0                   ;J
        TEST    BYTE PTR [BX],20H       ;]
        JE      $+5                     ;} 背景②が空間でなければ背景②を作成
        CALL    SSMP1                   ;J
MGPAT:  MOV     CX,MGPDT                ;]
MGPA2:  MOV     BX,OFFSET SPPAT-2       ;|
        ADD     BX,CX                   ;} 指定のスプライト・パターンを作成
        CALL    SPCH2                   ;|
MGVRT:  RET                             ;J
MGVDT   ENDP

MGV1W   PROC
        INC     BYTE PTR [MSPDT]        ;次の表示位置を更新（+1）する
        MOV     BX,MG1DT                ;]
        TEST    BYTE PTR [BX],04H       ;} 背景③がベタパターンならリターン
        JNE     MVRET                   ;J
        MOV     AL,CH                   ;]
        ROL     AL,1                    ;|
        MOV     CL,AL                   ;|
        MOV     CH,0                    ;} スプライト・パターンを作成
        MOV     BX,OFFSET SPPAT-2       ;|
        ADD     BX,CX                   ;|
        CALL    SPCH2                   ;|
MVRET:  RET                             ;J
MGV1W   ENDP

MGVD2   PROC
        MOV     BX,MG2DT                ;]
        TEST    BYTE PTR [BX],08H       ;} 背景②が空間（パターン無し）ならリターン
        JE      MPRET                   ;J
        PUSH    AX                      ;]
        CALL    SSMP2                   ;|
        POP     AX                      ;} 背景③のパターン作成
MPRET:  RET                             ;J
MGVD2   ENDP

SPMP2   PROC
        MOV     BX,OFFSET MAP02         ;]
        ADD     BX,CX                   ;|
        MOV     AL,[BX]                 ;|
        MOV     BX,OFFSET MA2TB         ;} 背景①のパターンを指定スプライトエリアに作成
SPMPS:  CALL    SPCHR                   ;|
        CALL    KPREG                   ;|
        RET                             ;J
SPMP2   ENDP

SGADR   PROC
        MOV     DX,4A0H                 ;]
        OUT     DX,AX                   ;|
        MOV     DI,CS:SGADT             ;|
```

```
        MOV     CX,16                      ;|
        REP     MOVSW                      ;| 指定アドレスにデータを転送
        ROL     AX,1                       ;|
        RET                                ;|
SGADR   ENDP

SPCHR   PROC
        ADD     AL,AL                      ;|
        MOV     DL,AL                      ;|
        MOV     DH,0                       ;|
        ADD     BX,DX                      ;|
SPCH2:  MOV     SI,[BX]                    ;|
        OUTPORT 4A4H,0C0CH                 ;| 指定のスプライトエリア(SGADR+1)に
        MOV     AX,0FFF0H                  ;| パターンを重ね合わせる
        CALL    SGADR                      ;|
        OUTPORT 4A4H,0CFCH                 ;|
        MOV     AX,0FFFEH                  ;|
        CALL    SGADR                      ;|
        CALL    SGADR                      ;|
        CALL    SGADR                      ;|
        CALL    SGADR                      ;|
        RET                                ;|
SPCHR   ENDP

SPMP1   PROC
        PUSH    CX                         ;|
        CALL    KPREG                      ;|
        POP     CX                         ;|
        MOV     BX,OFFSET MAP01            ;|
        ADD     BX,CX                      ;| 重ね合わせパターン(背景②)を指定スプライト
        MOV     AL,[BX]                    ;| エリアに作成
        MOV     BX,OFFSET MA1TB            ;|
        CALL    SPCHR                      ;|
        CALL    KPREG                      ;|
        RET                                ;|
SPMP1   ENDP

SFMP0   PROC
        PUSH    CX                         ;|
        CALL    KPREG                      ;|
        POP     CX                         ;|
        MOV     BX,OFFSET MAP00            ;|
        ADD     BX,CX                      ;| 背景①のパターンを指定スプライトエリアに作成
        MOV     AL,[BX]                    ;|
        MOV     BX,GPT00                   ;|
        CALL    SMFDI                      ;|
        RET                                ;|
SFMP0   ENDP

SSMP1   PROC
        TEST    BYTE PTR [BX],10H          ;|
        JNE     SFMP1                      ;| 背景②が重ね合わせパターンならSPMP1を実行
        CALL    SPMP1                      ;|
        JMP     SMPRT                      ;|
SFMP1:  PUSH    CX                         ;|
        CALL    KPREG                      ;|
        POP     CX                         ;|
        MOV     BX,OFFSET MAP01            ;|
        ADD     BX,CX                      ;| ベタパターン(背景②)を指定スプライトエリアに
        MOV     AL,[BX]                    ;| 作成
        SUB     AL,FPAT1                   ;|
        MOV     BX,GPT11                   ;|
        CALL    SMFDI                      ;|
```

```
SMPRT:  RET                                 ;」
SSMP1   ENDP

SSMP2   PROC
        TEST    BYTE PTR [BX],04H           ;┐
        CALL    KPREG                       ;│
        MOV     CX,MP2DT                    ;│
        JNE     SFMP2                       ;├ 背景③が重ね合わせパターンならSPMP2を実行
        CALL    SPMP2                       ;│
        JMP     SSMRT                       ;┘
SFMP2:  MOV     BX,OFFSET MAP02             ;┐
        ADD     BX,CX                       ;│
        MOV     AL,[BX]                     ;│
        SUB     AL,FPAT2                    ;├ ベタパターン(背景③)を指定スプライトエリアに
        MOV     BX,GPT21                    ;│ 作成
        CALL    SMFDI                       ;│
SSMRT:  RET                                 ;┘
SSMP2   ENDP

SMFDI   PROC
        MOV     AH,0                        ;┐
        SHL     AX,1                        ;│
        SHL     AX,1                        ;│
        SHL     AX,1                        ;│
        SHL     AX,1                        ;│
        SHL     AX,1                        ;│
        ADD     BX,AX                       ;│
        PUSH    DS                          ;├ 指定パターンを指定スプライトエリアに作成
        MOV     SI,BX                       ;│
        MOV     AX,BLUE                     ;│
        MOV     DS,AX                       ;│
        MOV     DX,4A4H                     ;│
        MOV     AX,28F0H                    ;│
        OUT     DX,AX                       ;│
        MOV     AX,0FFF0H                   ;│
        CALL    SGADR                       ;│
        POP     DS                          ;┘
        CALL    KPREG
        RET
SMFDI   ENDP

MGPDT   DW      0
MG1DT   DW      0
MSPDT   DB      1
MG2DT   DW      0
MP2DT   DW      0
SGADT   DW      0

MAKSG   ENDP
SPAT0   LABEL BYTE                          ;スプライト（ピーヨ）データ0
        DB      0,1FH,0,3FH,0,0FFH,1,0FFH
        DB      3,0FFH,7,0FFH,0FH,0FFH,0FH,0FFH
        DB      1FH,0FFH,1FH,0FFH,3FH,0FFH,3FH,0FFH
        DB      7FH,0FFH,7FH,0FFH,7FH,0FFH,7FH,0FFH
        DB      0,0,0,15H,0,3FH,0,7FH
        DB      1,0FFH,3,0FFH,7,0FFH,7,0FFH
        DB      0FH,0F9H,0FH,0F0H,1FH,0E0H,1FH,0E0H
        DB      3FH,0E0H,3FH,0E0H,3FH,0F0H,3FH,0F0H
        DB      0,0,0,0,0,0BH,0,0BH
        DB      0,05FH,0,07FH,1,79H,1,70H
        DB      2,0E6H,2,0EFH,5,0DFH,5,0DFH
        DB      0BH,0DCH,0BH,0DCH,5,0EFH,5,0EFH
        DB      0,0,0,15H,0,3FH,0,7FH
```

```
        DB      1,0FFH,3,0FFH,7,0F9H,7,0F0H
        DB      0FH,0EFH,0FH,0EFH,1FH,0DFH,1FH,0DFH
        DB      3FH,0DCH,3FH,0DCH,3FH,0EFH,3FH,0EFH
        DB      0,0,0,15H,0,34H,0,74H
        DB      1,0A0H,3,80H,6,80H,6,80H
        DB      0DH,9,0DH,0,1AH,0,1AH,0
        DB      34H,0,34H,0,3AH,0,3AH,0

SPAT1   LABEL BYTE                          ;スプライト（ピーヨ）データ1
        DB      0F0H,0,0F8H,0,0FEH,0,0FFH,0
        DB      0FFH,80H,0FFH,0C0H,0FFH,0E0H,0FFH,0E0H
        DB      0FFH,0F0H,0FFH,0F0H,0FFH,0F8H,0FFH,0F8H
        DB      0FFH,0F8H,0FFH,0F8H,0FFH,0F8H,0FFH,0F8H
        DB      0,0,50H,0,0F8H,0,0FEH,0
        DB      0FFH,0,0FFH,80H,0FFH,0C0H,0FFH,0C0H
        DB      0F9H,0E0H,0F0H,0E0H,60H,70H,60H,70H
        DB      60H,70H,60H,70H,0F0H,0F0H,0F0H,0F0H
        DB      0,0,0,0,40H,0,60H,0
        DB      0FCH,0,0FEH,0,0F9H,80H,0F0H,80H
        DB      66H,40H,6FH,40H,9FH,0A0H,9FH,0A0H
        DB      93H,0A0H,93H,0A0H,6FH,040H,6FH,40H
        DB      0,0,50H,0,0F8H,0,0FEH,0
        DB      0FFH,0,0FFH,80H,0F9H,0C0H,0F0H,0C0H
        DB      6FH,60H,6FH,60H,9FH,0B0H,9FH,0B0H
        DB      93H,0B0H,93H,0B0H,6FH,070H,6FH,70H
        DB      0,0,50H,0,0B8H,0,9EH,0
        DB      3,0,1,80H,0,40H,0,40H
        DB      9,20H,0,20H,0,10H,0,10H
        DB      0,10H,0,10H,0,30H,0,30H

SPAT2   LABEL BYTE                          ;スプライト（ピーヨ）データ2
        DB      7FH,0FFH,7FH,0FFH,7FH,0FFH,7FH,0FFH
        DB      3FH,0FFH,3FH,0FFH,5FH,0FFH,5FH,0FFH
        DB      0FFH,0FFH,0FFH,0FFH,7FH,0FFH,7FH,0FFH
        DB      3FH,0FFH,1FH,0FFH,0FH,0FFH,3,0FEH
        DB      3FH,0FFH,3FH,0FFH,3FH,0FFH,3FH,0FFH
        DB      1EH,4DH,1EH,4EH,0FH,26H,0FH,27H
        DB      47H,7,67H,7,33H,0FFH,39H,0FFH
        DB      1FH,0FFH,0FH,0FFH,2,0AAH,0,0
        DB      0BH,0F0H,0BH,0F9H,5,0FFH,5,0FFH
        DB      4,0EDH,4,0EEH,3,76H,3,77H
        DB      1,57H,1,57H,10H,7FH,10H,7FH
        DB      1,51H,5,55H,0,0,0,0
        DB      3FH,0F0H,3FH,0F9H,3FH,0FFH,3FH,0FFH
        DB      1EH,0EDH,1EH,0EEH,0FH,76H,0FH,77H
        DB      47H,57H,67H,57H,33H,0FFH,39H,0FFH
        DB      1FH,0FFH,0FH,0FFH,2,0AAH,0,0
        DB      34H,0,34H,0,3AH,0,3AH,0
        DB      1AH,0,1AH,0,0CH,0,0CH,0
        DB      46H,0,66H,0,23H,80H,29H,80H
        DB      1EH,0AEH,0AH,0AAH,2,0AAH,0,0

SPAT3   LABEL BYTE                          ;スプライト（ピーヨ）データ3
        DB      0FFH,0F8H,0FFH,0F8H,0FFH,0FCH,0FFH,0FCH
        DB      0FFH,0FEH,0FFH,0FEH,0FFH,0FFH,0FFH,0FFH
        DB      0FFH,0FFH,0FFH,0FFH,0FFH,3EH,0FCH,1CH
        DB      0F8H,0,0E0H,0,80H,0,0,0
        DB      0FFH,0F0H,0FFH,0F0H,0FFH,0F8H,0FFH,0F8H
        DB      0FDH,0D4H,0FBH,0D4H,3,0AAH,7,0AAH
        DB      0FFH,0,0FFH,0,0FCH,0,0F8H,0
        DB      0E0H,0,0C0H,0,0,0,0,0
        DB      0F0H,0E0H,0F9H,0E0H,0FFH,0D8H,0FFH,0D8H
        DB      0FDH,9CH,0FBH,9CH,3,3EH,6,3EH
```

```
        DB      0FEH,2AH,0FAH,2AH,50H,0,50H,0
        DB      0,0,0,0,0,0,0,0
        DB      0F0H,0F0H,0F9H,0F0H,0FFH,0F8H,0FFH,0F8H
        DB      0FDH,0DCH,0FBH,0DCH,3,0BEH,7,0BEH
        DB      0FFH,2AH,0FFH,2AH,0FCH,0,0F8H,0
        DB      0E0H,0,0C0H,0,0,0,0,0
        DB      0,10H,0,10H,0,20H,0,20H
        DB      0,40H,0,040H,0,80H,1,80H
        DB      1,0,5,0,0ACH,0,0A8H,0
        DB      0E0H,0,0C0H,0,0,0,0,0

SPAT4   LABEL BYTE                      ;スプライト（ピーヨ）データ4
        DB      7FH,0FFH,7FH,0FFH,7FH,0FFH,0FFH,0FFH
        DB      0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH
        DB      0FFH,0FFH,0FFH,0FFH,3FH,0FFH,3FH,0FFH
        DB      0FH,0FFH,0FH,0FEH,0,54H,0,0
        DB      3FH,0FFH,3FH,0FFH,3FH,0FFH,3FH,0FFH
        DB      1FH,6DH,1FH,6EH,47H,76H,47H,77H
        DB      3FH,73H,3FH,73H,7,69H,7,69H
        DB      0,0,0,0,0,0,0,0
        DB      0BH,0F0H,3,0F9H,5,0FFH,5,0FFH
        DB      5,2DH,5,2EH,1,56H,1,57H
        DB      1,7BH,1,7BH,0,7DH,0,7DH
        DB      0,54H,0,54H,0,0,0,0
        DB      3FH,0F0H,3FH,0F9H,3FH,0FFH,3FH,0FFH
        DB      1FH,6DH,1FH,6EH,47H,76H,47H,77H
        DB      3BH,7BH,3BH,7BH,7,7DH,7,7DH
        DB      0,54H,0,54H,0,0,0,0
        DB      34H,0,3CH,0,3AH,0,3AH,0
        DB      1AH,40H,1AH,40H,46H,20H,46H,20H
        DB      3AH,0,3AH,0,7,0,0,7
        DB      0,0,0,0,0,0,0,0

SPAT5   LABEL BYTE                      ;スプライト（ピーヨ）データ5
        DB      0FFH,0F8H,0FFH,0F8H,0FFH,0FCH,0FFH,0FCH
        DB      0FFH,0F8H,0FFH,0F8H,0FFH,0FCH,0FFH,0FCH
        DB      0FFH,0FCH,0FFH,0FCH,0FFH,0FEH,0FFH,0FEH
        DB      0F8H,0FEH,0,0FEH,0,54H,0,0
        DB      0FFH,0F0H,0FFH,0F0H,0FFH,0F0H,0FFH,0F0H
        DB      0FDH,0F0H,0FBH,0F0H,3,0F8H,7,0F8H
        DB      0FFH,68H,0FFH,68H,0FCH,54H,0F8H,54H
        DB      0,0,0,0,0,0,0,0
        DB      0F0H,0E0H,0F9H,0E0H,0FFH,0C0H,0FFH,0C0H
        DB      0FDH,0A0H,0FBH,0A0H,3,0B0H,7,0B0H
        DB      0FFH,38H,0FFH,38H,54H,7CH,50H,7CH
        DB      0,54H,0,54H,0,0,0,0
        DB      0F0H,0F0H,0F9H,0F0H,0FFH,0F0H,0FFH,0F0H
        DB      0FDH,0F0H,0FBH,0F0H,3,0F8H,7,0F8H
        DB      0FFH,78H,0FFH,78H,0FCH,7CH,0F8H,7CH
        DB      0,54H,0,54H,0,0,0,0
        DB      0,10H,0,10H,0,30H,0,30H
        DB      0,50H,0,50H,0,48H,0,48H
        DB      0,40H,0,40H,0A8H,0,0A8H,0
        DB      0,0,0,0,0,0,0,0

BKT11   LABEL BYTE                      ;背景②重ね合わせ・パターンデータ(パターン番号=1)
        DB      7,0E0H,7,0E0H,7,0E0H,7,0E0H
        DB      7,0E0H,7,0E0H,7,0E0H,7,0E0H
        DB      7,0E0H,7,0E0H,7,0E0H,7,0E0H
        DB      7,0E0H,7,0E0H,7,0E0H,7,0E0H
        DB      2,0,2,0,2,0,2,0
        DB      2,0,2,0,2,0,2,0
        DB      2,0,2,0,2,0,2,0
```

```
        DB      2,0,2,0,2,0,2,0
        DB      3,40H,3,0C0H,3,40H,3,0C0H
        DB      3,40H,3,0C0H,3,40H,3,0C0H
        DB      3,40H,3,0C0H,3,40H,3,0C0H
        DB      3,40H,3,0C0H,3,40H,3,0C0H
        DB      3,80H,3,0,3,80H,3,0
        DB      3,80H,3,0,3,80H,3,0
        DB      3,80H,3,0,3,80H,3,0
        DB      3,80H,3,0,3,80H,3,0
        DB      0,0,0,0,0,0,0,0
        DB      0,0,0,0,0,0,0,0
        DB      0,0,0,0,0,0,0,0
        DB      0,0,0,0,0,0,0,0

BKT21   LABEL BYTE                  ;背景③重ね合わせ・パターンデータ (パターン番号=1)
        DB      0,0,0,0,1,80H,1,80H
        DB      3,0C0H,3,0C0H,7,0E0H,7,0E0H
        DB      0FH,0F0H,0FH,0F0H,1FH,0F8H,1FH,0F8H
        DB      3FH,0FCH,3FH,0FCH,7FH,0FEH,7FH,0FEH
        DB      0,0,0,0,0,0,0,0
        DB      1,80H,1,80H,3,0C0H,3,0C0H
        DB      7,0E0H,7,0E0H,0FH,0F0H,0FH,0F0H
        DB      1FH,0F8H,1FH,0F8H,3FH,0FCH,3FH,0FCH
        DB      0,0,0,0,0,0,0,0
        DB      0,0,0,0,2,0,2,0
        DB      2,0,2,0,0AH,0,0AH,0
        DB      0AH,0,0AH,0,2AH,0,2AH,0
        DB      0,0,0,0,0,0,0,0
        DB      1,80H,1,80H,3,40H,3,40H
        DB      7,0A0H,7,0A0H,0FH,50H,0FH,50H
        DB      1FH,0A8H,1FH,0A8H,3FH,54H,3FH,54H
        DB      0,0,0,0,0,0,0,0
        DB      0,0,0,0,0,0,0,0
        DB      0,0,0,0,0,0,0,0
        DB      0,0,0,0,0,0,0,0

BKD00   LABEL BYTE                  ;背景① ベタ・パターンデータ (パターン番号=0)
        DB      0FFH,0FFH,0FFH,0FFH,0F7H,0F7H,0F7H,0F7H
        DB      0DDH,0DDH,0DDH,0DDH,33H,33H,33H,33H
        DB      0AAH,0AAH,0AAH,0AAH,44H,44H,44H,44H
        DB      10H,10H,10H,10H,0,0,0,0

BKD01   LABEL BYTE                  ;背景① ベタ・パターンデータ (パターン番号=1)
        DB      0,0,0,0,10H,10H,10H,10H
        DB      44H,44H,44H,44H,0AAH,0AAH,0AAH,0AAH
        DB      33H,33H,33H,33H,0DDH,0DDH,0DDH,0DDH
        DB      0F7H,0F7H,0F7H,0F7H,0FFH,0FFH,0FFH,0FFH

BKD12   LABEL BYTE                  ;背景② ベタ・パターンデータ (パターン番号=2)
        DB      6AH,20H,6AH,20H,50H,20H,50H,20H
        DB      68H,20H,68H,20H,50H,0,50H,0
        DB      6CH,0,6CH,0,54H,0,54H,0
        DB      6CH,0,6CH,0,40H,0,40H,0
        DB      7FH,0DCH,7FH,0DCH,7FH,0DAH,7FH,0DAH
        DB      7FH,0C4H,7FH,0C4H,55H,52H,55H,52H
        DB      7BH,0F4H,7BH,0F4H,7BH,0FAH,7BH,0FAH
        DB      7BH,0E4H,7BH,0E4H,55H,52H,55H,52H
        DB      7FH,8AH,7FH,8AH,75H,4,75H,4
        DB      6AH,09AH,6AH,09AH,50H,0,50H,0
        DB      6AH,8AH,6AH,8AH,72H,4,72H,4
        DB      78H,09AH,78H,09AH,50H,0,50H,0
        DB      0,0,0,0,0,0,0,0
        DB      0,0,0,0,0,0,0,0
```

```
        DB      0,0,0,0,0,0,0,0
        DB      0,0,0,0,0,0,0,0

BKD22   LABEL BYTE                          ;背景③ベタ・パターンデータ（パターン番号=2)
        DB      6FH,0FFH,6FH,0FFH,6FH,0FCH,6FH,0FCH
        DB      6FH,0FFH,6FH,0FFH,6FH,0FCH,6FH,0FCH
        DB      6FH,0FFH,6FH,0FFH,6FH,0FCH,6FH,0FCH
        DB      6FH,0FFH,6FH,0FFH,6FH,0FFH,6FH,0FFH
        DB      2,0ABH,2,0ABH,2,0ABH,2,0ABH
        DB      2,0ABH,2,0ABH,2,0ABH,2,0ABH
        DB      2,0ABH,2,0ABH,2,0ABH,2,0ABH
        DB      2,0ABH,2,0ABH,0,0,0,0
        DB      5BH,0BFH,5BH,0BFH,5BH,0BFH,5BH,0BFH
        DB      5BH,0BFH,5BH,0BFH,5BH,0BFH,5BH,0BFH
        DB      5BH,0BFH,5BH,0BFH,5BH,0BFH,5BH,0BFH
        DB      5BH,0BFH,5BH,0BFH,11H,15H,11H,15H
        DB      0,0,0,0,0,0,0,0
        DB      0,0,0,0,0,0,0,0
        DB      0,0,0,0,0,0,0,0
        DB      0,0,0,0,0,0,0,0

BKD23   LABEL BYTE                          ;背景③ベタ・パターンデータ（パターン番号=3)
        DB      3FH,0F6H,3FH,0F6H,0FFH,0F6H,0FFH,0F6H
        DB      3FH,0F6H,3FH,0F6H,0FFH,0F6H,0FFH,0F6H
        DB      3FH,0F6H,3FH,0F6H,0FFH,0F6H,0FFH,0F6H
        DB      3FH,0F6H,3FH,0F6H,0FFH,0F6H,0FFH,0F6H
        DB      0D5H,40H,0D5H,40H,0D5H,40H,0D5H,40H
        DB      0D5H,40H,0D5H,40H,0D5H,40H,0D5H,40H
        DB      0D5H,40H,0D5H,40H,0D5H,40H,0D5H,40H
        DB      0D5H,40H,0D5H,40H,0,0,0,0
        DB      0FDH,0DAH,0FDH,0DAH,0FDH,0DAH,0FDH,0DAH
        DB      0FDH,0DAH,0FDH,0DAH,0FDH,0DAH,0FDH,0DAH
        DB      0FDH,0DAH,0FDH,0DAH,0FDH,0DAH,0FDH,0DAH
        DB      0FDH,0DAH,0FDH,0DAH,0A8H,88H,0A8H,88H
        DB      0,0,0,0,0,0,0,0
        DB      0,0,0,0,0,0,0,0
        DB      0,0,0,0,0,0,0,0
        DB      0,0,0,0,0,0,0,0

ALMAP   PROC
        MOV     DI,OFFSET CKMAP         ;
        MOV     BX,OFFSET MAP01         ;
        MOV     SI,OFFSET MAP02         ;
        MOV     CX,DMAXX*DMAXY          ;
ALMAL:  MOV     BYTE PTR [DI],1         ;
        MOV     AL,[BX]                 ;
        OR      AL,AL                   ;
        JE      ALMP1                   ;
        OR      BYTE PTR [DI],20H       ;
        CMP     AL,FPAT1                ;
        JB      ALMP1                   ;
        OR      BYTE PTR [DI],10H       ;  初期画面作成用に描き換え情報マップを作成
ALMP1:  MOV     AL,[SI]                 ;
        OR      AL,AL                   ;
        JE      ALMP2                   ;
        OR      BYTE PTR [DI],08H       ;
        CMP     AL,FPAT2                ;
        JB      ALMP2                   ;
        OR      BYTE PTR [DI],04H       ;
ALMP2:  INC     DI                      ;
        INC     BX                      ;
        INC     SI                      ;
        LOOP    ALMAL                   ;
```

```
        RET                                     ;J
ALMAP   ENDP

NOMP1   PROC
        MOV     BX,OFFSET CKMAP                 ;]
        MOV     DI,OFFSET MAP01                 ;
        MOV     CX,DMAXX*DMAXY                  ;
NOM1L:  MOV     BYTE PTR [BX],0                 ;
        MOV     AL,[DI]                         ;
        OR      AL,AL                           ;
        JE      NOM11                           ;  背景②非スクロール時の描き換え情報マップを
        OR      BYTE PTR [BX],20H               ;  作成
        CMP     AL,FPAT1                        ;
        JB      NOM11                           ;
        OR      BYTE PTR [BX],10H               ;
NOM11:  INC     BX                              ;
        INC     DI                              ;
        LOOP    NOM1L                           ;
        RET                                     ;J
NOMP1   ENDP

MVMP1   PROC
        MOV     BX,OFFSET CKMAP                 ;]
        MOV     SI,OFFSET MAP01                 ;
        MOV     DI,OFFSET MAP1S                 ;
        MOV     CX,DMAXY                        ;
        PUSH    ES                              ;
        MOV     AX,DS                           ;
        MOV     ES,AX                           ;
        REP     MOVSB                           ;
        POP     ES                              ;
        MOV     SI,OFFSET MAP01                 ;
        MOV     DI,OFFSET MAP01+DMAXY           ;
        MOV     CX,DMAXX*DMAXY                  ;
MV1LP:  MOV     AL,[DI]                         ;
        CMP     AL,[SI]                         ;
        MOV     [SI],AL                         ;  背景②スクロール時の描き換え情報マップを
        MOV     BYTE PTR [BX],0                 ;
        JE      MVM11                           ;  作成
        OR      BYTE PTR [BX],01H               ;
MVM11:  OR      AL,AL                           ;
        JE      MVM12                           ;
        OR      BYTE PTR [BX],20H               ;
        CMP     AL,FPAT1                        ;
        JB      MVM12                           ;
        OR      BYTE PTR [BX],10H               ;
MVM12:  INC     BX                              ;
        INC     SI                              ;
        INC     DI                              ;
        LOOP    MV1LP                           ;
        RET                                     ;J
MVMP1   ENDP

MVMP2   PROC
        MOV     BX,OFFSET CKMAP                 ;]
        MOV     SI,OFFSET MAP02                 ;
        MOV     DI,OFFSET MAP2S                 ;
        MOV     CX,DMAXY                        ;
        PUSH    ES                              ;
        MOV     AX,DS                           ;
        MOV     ES,AX                           ;
        REP     MOVSB                           ;
        POP     ES                              ;]
```

```
        MOV     SI,OFFSET MAP02               ;)
        MOV     DI,OFFSET MAP02+DMAXY         ;
        MOV     CX,DMAXX*DMAXY                ;
MV2LP:  MOV     AL,[DI]                       ;
        CMP     AL,FPAT2                      ; 背景③スクロール時の描き換え情報マップを
        JB      MVM20                         ; 作成
        AND     BYTE PTR [BX],0FEH            ;
        OR      BYTE PTR [BX],04H             ;
MVM20:  CMP     AL,[SI]                       ;
        MOV     [SI],AL                       ;
        JE      MVM21                         ;
        OR      BYTE PTR [BX],01H             ;
MVM21:  OR      AL,AL                         ;
        JE      MVM22                         ;
        OR      BYTE PTR [BX],08H             ;
MVM22:  INC     BX                            ;
        INC     SI                            ;
        INC     DI                            ;
        LOOP    MV2LP                         ;)
        RET
MVMP2   ENDP

                                              ;各スプライト（フラグ,X,Y)
SPWOK   DB      SPVAL*3 DUP(0)                ;各スプライト（パターンアドレス)
SPPAT   DB      SPVAL*2 DUP(0)                ;各スプライト表示順位
SLANK   DB      SPVAL   DUP(0)

QSSET   PROC                                  ;)
        MOV     SI,OFFSET SPWOK               ;
        MOV     CX,SPVAL                      ;
QSTSL:  SHR     BYTE PTR [SI],1               ;
        JNB     QSSL1                         ;
        XOR     AX,AX                         ;
        MOV     BX,AX                         ;
        MOV     AL,[SI+1]                     ;
        MOV     BL,[SI+2]                     ;
        SHL     AX,1                          ; 使用中のスプライトの座標にあたる描き換え
        SHL     AX,1                          ; 情報マップを作成
        SHL     AX,1                          ;
        SHL     AX,1                          ;
        ADD     AX,OFFSET CKMAP               ;
        ADD     BX,AX                         ;
        OR      BYTE PTR [BX],01H             ;
QSSL1:  ADD     SI,3                          ;
        LOOP    QSTSL                         ;
        RET                                   ;)
QSSET   ENDP

PIYOM   PROC                                  ;)
        MOV     BP,OFFSET SPWOK               ;
        MOV     SI,OFFSET PIYO0               ;
        MOV     DX,OFFSET SPPAT               ;
        CALL    PMOVS                         ; 各ピーヨを適当に動かす
        CALL    PMOVS                         ;
        CALL    PMOVS                         ;
        CALL    PMOVS                         ;
        CALL    PMOVS                         ;)
        RET

PMOVS   PROC                                  ;)
        DEC     BYTE PTR [SI+2]               ;
        JNE     $+5                           ;
        CALL    PDCHN                         ;)
        CALL    PTCHN
```

```
        MOV     CH,[SI+0]               ;
        MOV     CL,[SI+1]               ;
        MOV     AL,[SI+3]               ;
        CMP     AL,1                    ;
        JB      BCSTA                   ;
        JE      PDCK1                   ;
        CMP     AL,3                    ;
        JB      PDCK2                   ;
        JE      PDCK3                   ;
PDCK4:  DEC     CH                      ;
        JNE     $+5                     ;
        CALL    PDCHN                   ;
        JMP     BCSTA                   ;
PDCK1:  CALL    BICHK                   ;
        INC     CL                      ;
        CMP     CL,DMAXY-2              ;
        JB      BCSTA                   ;
        DEC     CL                      ;
        JMP     BCSTA                   ;
PDCK2:  CALL    BICHK                   ;
        JMP     BCSTA                   ;
PDCK3:  CALL    BICHK                   ;
        DEC     CL                      ;
        JNS     BCSTA                   ;
        INC     CL                      ;
BCSTA:  MOV     [SI+0],CH               ;
        MOV     [SI+1],CL               ;
        XCHG    BP,BX                   ; ピーヨを適当に動かす
        MOV     AH,CH                   ;
        MOV     AL,CL                   ;
        INC     AH                      ;
        INC     AL                      ;
        MOV     BYTE PTR [BX],1         ;
        INC     BX                      ;
        MOV     [BX],CH                 ;
        INC     BX                      ;
        MOV     [BX],CL                 ;
        INC     BX                      ;
        MOV     BYTE PTR [BX],1         ;
        INC     BX                      ;
        MOV     [BX],AH                 ;
        INC     BX                      ;
        MOV     [BX],CL                 ;
        INC     BX                      ;
        MOV     BYTE PTR [BX],1         ;
        INC     BX                      ;
        MOV     [BX],CH                 ;
        INC     BX                      ;
        MOV     [BX],AL                 ;
        INC     BX                      ;
        MOV     BYTE PTR [BX],1         ;
        INC     BX                      ;
        MOV     [BX],AH                 ;
        INC     BX                      ;
        MOV     [BX],AL                 ;
        INC     BX                      ;
        XCHG    BP,BX                   ;
        ADD     SI,5                    ;
        RET                             ;

BICHK   PROC
        INC     CH                      ;
        MOV     AL,CH                   ;
```

```
         CMP     AL,DMAXX-1              ;
         JB      BICRT                   ;
         DEC     CH                      ;
         CALL    RND                     ;
         OR      AL,00010000B            ;  画面右端へ到達した場合には最低16回以
         AND     AL,00011111B            ;  後退するようにする
         MOV     [SI+2],AL               ;
         MOV     BYTE PTR [SI+3],4       ;
BICRT:   RET                             ;
BICHK    ENDP

PDCHN    PROC
         CALL    RND                     ;
         OR      AL,00000010B            ;
         AND     AL,00001111B            ;
         MOV     [SI+2],AL               ;  方向を変更し、カウンタ値を設定
         CALL    RND                     ;
         AND     AL,00000011B            ;
         MOV     [SI+3],AL               ;
         RET                             ;
PDCHN    ENDP

PTCHN    PROC
         XOR     BYTE PTR [SI+4],1       ;
         MOV     BX,OFFSET CHR00         ;
         JE      PMPTS                   ;
         MOV     BX,OFFSET CHR01         ;
PMPTS:   PUSH    ES                      ;
         MOV     CX,CS                   ;
         MOV     ES,CX                   ;
         MOV     CX,4                    ;  ピーヨ番号を変更する
         XCHG    DI,DX                   ;
         XCHG    SI,BX                   ;
         REP     MOVSW                   ;
         XCHG    DI,DX                   ;
         XCHG    SI,BX                   ;
         POP     ES                      ;
         RET                             ;
PTCHN    ENDP
PMOVS    ENDP

PIY00    DB      2,3,1,0,0               ;
PIY01    DB      10,6,1,0,1              ;
PIY02    DB      15,11,1,0,0             ;  ピーヨ・0〜4 (X,Y,カウンタ,方向,ピーヨ番号)
PIY03    DB      22,12,1,0,0             ;
PIY04    DB      8,14,1,0,0              ;

CHR00    DW      SPAT0,SPAT1,SPAT2,SPAT3 ;ピーヨ番号=0に用いるパターン
CHR01    DW      SPAT0,SPAT1,SPAT4,SPAT5 ;ピーヨ番号=1に用いるパターン

PIYOM    ENDP

RND      PROC
         MOV     AL,RNDDT                ;
         MOV     AH,AL                   ;
         ADD     AL,AL                   ;
         ADD     AL,AL                   ;
         ADD     AL,AH                   ;  乱数発生ルーチン
         INC     AL                      ;
         MOV     RNDDT,AL                ;
         RET                             ;
RNDDT    DB      10H                     ;
RND      ENDP
```

```
KPREG   PROC
        XCHG    BX,CS:KPRBX         ;
        XCHG    CX,CS:KPRCX         ;
        XCHG    DX,CS:KPRDX         ;
        RET                         ; レジスターを一時保存する
KPRBX   DW      0                   ;
KPRCX   DW      0                   ;
KPRDX   DW      0                   ;
KPREG   ENDP

CKMAP   DB      DMAXX*DMAXY DUP(0)  ;描き換え情報マップエリア
MAP00   LABEL BYTE                  ;背景①マップ
        DB      0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1
        DB      0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1
        DB      0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1
        DB      0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1
        DB      0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1
        DB      0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1
        DB      0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1
        DB      0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1
        DB      0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1
        DB      0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1
        DB      0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1
        DB      0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1
        DB      0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1
        DB      0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1
        DB      0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1
        DB      0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1
        DB      0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1
        DB      0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1
        DB      0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1
        DB      0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1
        DB      0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1
        DB      0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1
        DB      0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1
        DB      0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1
        DB      0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1
        DB      0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1
        DB      0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1
        DB      0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1
        DB      0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1
        DB      0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1
        DB      0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1
        DB      0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1

MAP01   LABEL BYTE                  ;背景②マップ
        DB      0,0,0,0,0,0,0,2,2,2,2,2,2,2,2,2
        DB      0,0,0,0,0,0,0,2,1,2,1,2,1,2,1,2
        DB      0,0,0,0,0,0,0,2,1,2,1,2,1,2,1,2
        DB      0,0,0,0,0,0,0,2,1,2,1,2,1,2,1,2
        DB      0,0,0,0,0,0,0,2,1,2,1,2,1,2,1,2
        DB      0,0,0,0,0,0,0,2,2,2,2,2,2,2,2,2
        DB      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
        DB      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
        DB      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
        DB      0,0,0,0,0,0,2,1,1,2,2,2,2,2,2,2
        DB      0,0,0,2,1,1,2,0,0,2,0,0,2,2,2,2
        DB      0,0,0,0,0,0,2,1,1,2,2,2,2,2,2,2
        DB      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
        DB      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
        DB      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
        DB      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
        DB      1,2,1,2,1,2,1,2,1,2,1,2,1,2,1,2
```

```
        DB      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
        DB      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
        DB      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
        DB      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
        DB      0,0,0,0,0,0,0,0,0,0,0,0,2,2,2,2
        DB      0,0,0,0,0,0,0,0,0,0,0,2,2,2,2,2
        DB      0,0,0,0,0,0,2,1,1,2,1,1,2,2,2,2
        DB      0,0,0,0,0,0,2,1,1,2,1,1,2,1,1,2
        DB      0,0,0,0,0,0,2,1,1,2,1,1,2,1,1,2
        DB      0,0,0,0,0,0,2,1,1,2,1,1,2,1,1,2
        DB      0,0,0,0,0,0,2,1,1,2,1,1,2,2,2,2
        DB      0,0,0,0,0,0,0,0,0,0,0,2,2,2,2,2
        DB      0,0,0,0,0,0,0,0,0,0,0,0,2,2,2,2
        DB      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
        DB      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0

MAP1S   LABEL BYTE                          ;背景②ループ用サブエリア
        DB      16 DUP(0)

MAP02   LABEL BYTE                          ;背景③マップ
        DB      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
        DB      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
        DB      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
        DB      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
        DB      0,0,0,0,0,0,1,2,2,2,2,2,2,2,2,2
        DB      0,0,0,0,0,0,1,3,3,3,3,3,3,3,3,3
        DB      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
        DB      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
        DB      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
        DB      0,0,0,0,0,0,0,0,0,0,0,0,1,2,2,2
        DB      2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2
        DB      3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3
        DB      0,0,0,0,0,0,0,0,0,0,0,0,1,3,3,3
        DB      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
        DB      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
        DB      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
        DB      0,0,0,1,2,2,2,2,2,2,2,2,2,2,2,2
        DB      0,0,0,1,3,3,3,3,3,3,3,3,3,3,3,3
        DB      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
        DB      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
        DB      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
        DB      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
        DB      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
        DB      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
        DB      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
        DB      0,0,0,0,0,0,0,0,0,0,0,0,0,1,2,2
        DB      0,0,0,0,0,0,0,0,0,0,0,0,0,1,2,2
        DB      0,0,0,0,0,0,0,0,0,0,0,0,0,1,3,3
        DB      0,0,0,0,0,0,0,0,0,0,0,0,0,1,3,3
        DB      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
        DB      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
        DB      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
MAP2S   DB      16 DUP(0)

TXCLR   LABEL BYTE                          ;テキスト・クリア & 文字の属性の保存
        DB      1BH,"[2J"
        DB      1BH,"[s",1BH,"[>5h",1BH,"[1>h$"

TKEEP   LABEL BYTE                          ;文字の属性の復元
        DB      1BH,"[u",1BH,"[>5l",1BH,"[1>l$"

CODE    ENDS
```

```
STACK   SEGMENT STACK                   ;スタック・セグメントの定義
        DW      100H    DUP(0)
STACK   ENDS

TEXT    SEGMENT AT 0A000H               ;テキスト・コードセグメントの定義
TEXT    ENDS

BLUE    SEGMENT AT 0A800H               ;B面のセグメントの定義
BLUE    ENDS

        END
```

サンプルですから、マップもパターンも最低限しか用意していませんが、なんとなくゴーストタウンを5匹の『おばけのピーヨ』が飛び回っているという雰囲気が感じられたと思います。シフトキーを押すと、画面は一時停止します。ソフト・スプライトによって背景とキャラクタ、あるいはキャラクタとキャラクタとの重ね合わせも完璧に行われていることがわかります。スクロールは、動かない背景①、1回おきに動く背景②、毎回動く背景③の3重スクロールです。

それぞれのマップは1面分しか用意してないので、データをグルグル回してスクロールさせています。横スクロールの場合、部分描き換えを縦に行ったほうがブレが少なくなると『**4−1　スクロールとキャラクタ**』で書きましたが、そのためにはデータ自体を縦に並べたほうが好都合です。

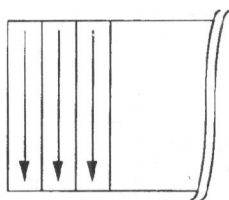



図 4-4-1　マップデータの並びと描き換えの順序

ここで、マップデータの内容についても触れておきましょう。背景①（MAP00）は一番下に位置していますからすべてベタ(透明データが不要)ですが、背景②と③（MAP01 と MAP02）のパターンは3種類のタイプに分けることができます。

◆全透明　：空間を意味し下のパターンが出る＝データ不要
◆部分透明：部分的に下のパターンが出る＝透明データ＋BRGI データ
◆ベタ　　：下のパターンは見えない＝BRGI データ

これらはパターン番号で区別されており、全透明はパターン番号＝０、部分透明とベタはラベル（背景①＝ FPAT1、背景②＝ FPAT2）で区別しています。ここではパターン数が少ないため、いずれも2以降がベタとなっています。ベタ・パターンは、それより下位にあるパターンを表示しなくていいので、表示速度が速い上にデータ量が少ない（透明データ不要）というメリットがあります。さらに、ベタ・パターンをG.VRAMの余りに用意しておけば（本プログラムでは 7D00H以降を利用）、4面同時にデータを転送することができます。

ただし、画面に直接転送して作画するのは、ベタ・パターンの上に重なるパターンがない場合で、上に別のマップが重なる場合には、一旦 7D00H以降にデータを作成してから転送します。このほうが速度効率がいいことは『**2－5　グラフィックVRAMの余り**』で説明したとおりです。また、キャラクタを表示しなければならない場合には、ソフト・スプライト処理として、これとは別にG.VRAMの一部（本プログラムでは 7D00H－32×SPVAL 以降）に合成したパターンを前もって作成しておきます。こうすると、描き換えに要する時間が短縮され、スクロールがよりきれいになるからです。もちろん、このエリアは通常のG.VRAMですから、テキスト画面によってマスクをしておく必要があります。

したがって、部分描き換えと一口にいっても、単に前回と違うパターンを描くだけではなく、その内容によって処理方法を違えたり準備も必要ということです。この例ではキャラクタを背景②と③の間に置いていますから、実際には次のような描き換えのタイプが存在しています。

	背景①	背景②	キャラクタ	背景③
直接表示（４面転送）	☆ベタ — —	無 ☆ベタ —	無 無 無	無 無 ☆ベタ
７D00Hに作成後転送	☆ベタ ☆ベタ ☆ベタ —	☆ベタ 無 ☆合成 ☆ベタ	無 無 無 無	無 ☆合成 ☆合成 ☆合成
ソフトスプライト （データを事前に作成）	☆ベタ ☆ベタ ☆ベタ ☆ベタ — — —	無 無 ☆合成 ☆合成 ☆ベタ ☆ベタ —	☆合成 ☆合成 ☆合成 ☆合成 ☆合成 ☆合成 —	無 ☆合成 無 ☆合成 無 ☆合成 ☆ベタ

☆　：作画の必要なパターン
—　：作画の不必要なパターン
無　：パターンが存在しない／空間
合成：重ね合わせの必要なパターン

これらは描き換えの際にすぐ判別できるよう、予め調べておくことが大切です。さもないと、描き換え時間が長くなりスクロールが汚くなってしまいます。そのため、こういった描き換え情報を示す専用のマップ（CKMAP）を用意し、ソフト・スプライトのデータ同様事前に作成しておきます。その内容は**図 4-4-2**のようなものです。

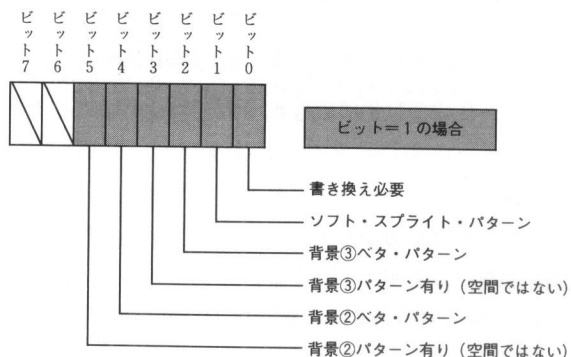



図 4-4-2　描き換え情報データ

データはスクロールの有無にかかわらず毎回更新します。というのは、ソフト・スプライトの情報もこの中に含まれるからです。更新は、背景②→背景③の順で行い、描き換え不要の場合でもその他のチェック（ベタかどうか／パターンの有無）をしておかなければなりません。次に、前回ソフト・スプライトのあった座標はすべて描き換え必要の判定をします。ここでキャラクタ移動（座標変化など）の処理をし、新たな座標について描き換え必要及びソフト・スプライトの認定をします。そして、ソフト・スプライト用のデータを作成すれば、すべての情報は整ったことになり、それを基に必要な表示を行えばいいわけです。

しかし、このソフト・スプライト用データ作成という作業は、簡単そうで意外と面倒なものです。今回は20のソフト・スプライトを利用できるようにしていますが、キャラクタの座標が変化するたびに当然この座標も変化します。一方、描き換えは図 4-4-1の順序で行いますから、データは表示に合わせた並びで作成しておく必要があります。プログラム中にある MLANKが座標に順位を付けるルーチンで、データ作成はそこで付けられた順位を参照しながら行うようになっています。

ここではすべてのソフト・スプライトを利用していますが、常にすべてが利用されるとは限りません。そこで、利用しないものの順位は０としてデータ作成などから除外するようにしています。また、同じ座標に複数のソフト・スプライトがあった場合には、番号の若いほうが上位に表示されるようプログラムで優先順位を付けています。

キャラクタの動きなどについては簡単ですから特に解説しませんが、移動方向は０＝停止、１＝右下、２＝右、３＝右上、４＝左となっており、乱数によって適当に動かしています。実際のゲームでは巨大キャラなども登場しますが、こういう場合にはソフト・スプライト側でもベタ・パターンと重ね合わせパターンとに分けるようにするなど、まだまだプログラムには発展の余地があります。この際ですから、とことんそういった可能性を追求してみてください。

【マップの裏話】

画面を構成するマップはデータを大量に消費します。一概にマップが広ければいいとは思いませんが、マップが広大になればゲームも大きくできるのは事実です。本章においてマップデータを工夫するアイデアを解説したのも、このような理由からです。

ところで、よくゲームの広告に画面数（マップ数ではない）が何万と書いてあるものがあります。まともにこんなことをしたら、それだけでディスクが10枚以上必要になってしまいます。実はこれ、同じ構成のマップをアチコチで使うということなのです。例えばの話、４画面で構成されているマップを10回利用すれば、40画面……となるわけです。

個人的にはこういうマップは嫌いですし、データの工夫とは認められません。第一、同じ構成のマップを違う場所と判断しなければならないなんて不自然です。トラップとして利用したり、距離感を出すために数回ループする程度でないと、プレイ中にイライラしてしまいそう……。こういった計算が成立するなら、先ほどのサンプル・スクロールなんて画面数は無限ですからネ。

【スポット処理】

テキスト画面を用いたスポット処理を第３章で紹介しましたが、グラフィック画面だけで完全なスポット処理を実現しているゲームも見かけます。難しそうですが、多重スクロールの背景②の役目をもう一度思いだしてください。

背景②は下にあるパターンの上に合成するわけですから、これを黒いパターンだけで作れば簡単にスポットを付けることができますね。それどころか、星型でもハート型でも構わないし、スポットの周囲をボカすような本格的スポットだって可能なのです。

な～ンだ……なんてバカにしてはいけません。こういったことは、手品と同じでタネを知る前に考えてこそ価値があるのですから……。

【１バイト以下のスクロール】

最後の節の多重スクロールでは１回おきに動く背景②というのがありましたが、これを毎回１バイト単位で動かすとスクロールはより美しくなります。このような場合、マップを小さなパターン（８×８ドット）にしたのではマップデータも２倍になり処理も大変です。そこで、プログラムでマップを半分ずつ（右半分と右隣りの左半分）を表示できるようにしておくと、毎回１バイト単位でスクロールさせることができます。

プログラムは常に進化していないと飽きられてしまいます。多重スクロールの基本を理解したなら、そこに新たなアイデアを盛り込むのは当然のこと。それがプログラムの個性です。本書のプログラムがどのように飛躍するのか……、マシン語の世界には夢とロマンがあふれているのです。

4－5　GDCによるスムーズ・スクロール

部分描き換えとEGCを駆使した多重スクロール処理はいかがでしたか。あるいは、これで十分とばかりに満足してしまったかもしれません。しかし、98シリーズにおけるスクロール処理には、もう1つ忘れてはならない方法があります。そうです。あの98版『スカイ・ブルーザー』で用いた GDC (Graphic Display Controller) による画面分割スクロールです。

GDCとは初代のPC-9801から用いられているグラフィックス用のLSIのことで、ライン、サークル、ボックス、ボックスペイント、ドットセット／リセット……等、グラフィックス処理を高速に実行する機能を有しています。かつては、ワイヤーフレームによるフライトシミュレータなどで、このGDCの機能が大いに活躍したものです。最近の機種では、GDCとEGCを組み合わせて、さらに高速な画面描写を実現しています。

GDCはメインCPUから独立した存在ですが、コマンドやパラメータを送るだけでコントロールできるようになっています。I／O ポートは、A0H／A2Hを使用します。GDCは、同じグラフィック画面を扱っているにもかかわらず、割り付けてあるG.VRAMアドレスは CPU側とは異なっています（図 4-5-1）。

	GDC	CPU
ブルー面	4000H ～7FFFH	A8000H～AFFFFH
レッド面	8000H ～BFFFH	B0000H～B7FFFH
グリーン面	C000H ～FFFFH	B8000H～BFFFFH
輝度面	0000H ～3FFFH	E0000H～E7FFFH

図 4-5-1

ご覧のように、GDCが管理しているG.VRAMのメモリ空間は、CPUのちょうど半分です。これは、GDCのアクセス単位がワード（2バイト）となっているためです。つまり、GDCにおいてG.VRAMアドレスが1違うと、画面上では2バイトの違いとなるわけです。メインCPU で扱うG.VRAMアドレスを基準にすると、GDCは偶数番地から2バイト単位でアクセスすることしかできないということです。

なお、CPUがG.VRAMをアクセスする場合、GDC がグラフィックス画面に描写中は避けなければなりません。GDCはコマンドによって実行されますから、このタイミングは必然的に CPU側で取ることになります。そのためには、GDCの実行状態を把握する必要があります。これは、ポートA0Hより得られるGDCのステータス・フラグを見ることで解決できます。図 4-5-2を見てください。ステータス・フラグのビット3がGDC描写中を示しています（b3＝1で描写中）。

ビット	内 容
0	DATA READY
1	FIFO BUFFER FULL
2	FIFO BUFFER EMPTY
3	DRAWING
4	DMA EXECUTE
5	VERTICAL SYNC
6	HORIZONTAL BLANK
7	LIGHT PEN DETECT

図 4-5-2 GDCステータス・フラグ

また、GDC内部には16バイトのFIFO (First In First Out) バッファがあり、書き込まれたコマンドやパラメータを一旦このバッファに蓄えて、随時読み出しては命令を実行しています。したがって、このバッファの状態にも注意を払わなければなりません。すなわち、バッファが空いているときにコマンドやパラメータを送るわけです。これは、ステータス・フラグのビット2で調べることができます (b2＝1でエンプティ)。

では、実際に GDCを使ったスクロールを行ってみましょう。ラインやサークルなどについては、いろいろな本で頻繁に紹介されていますので本書では省きます。このスクロールは、EGC を利用したソフトウェア・スクロールとは異なり、いわゆるハードウェア・スクロールです。すなわち、G.VRAMの内容は全く変化させずに、画面の表示開始位置をずらすことによって、見かけ上スクロールしているように見せています。

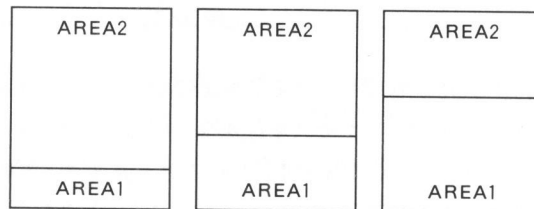



図 4-5-3

図 4-5-3を見てください。GDC を使うと、このようにグラフィック画面を任意のアドレス(ワード単位)で2分割して表示することができるのです。この時、B,R,G,I各プレーンに対するオフセットアドレスの0が表示される位置は、常にAREA1の左上ということになります。ここで境界を徐々にずらしていけば、画面はスクロールしたように見えます。これが、ハードウェア・スクロールの実体です。

GDC コントロールは、出力ポートからコマンドとパラメータを送って行います。コマンド用のポートはA0H、パラメータ用のポートはA2Hです。スクロール・コマ

ンドは 70Hとなっています。

さて、コマンドの次に送るパラメータですが、これは少々複雑です。まず、画面を図 4-5-4のようにAREA1、AREA2に2分割して、各分割画面の開始アドレスとライン数をそれぞれSAD1、SL1とSAD2、SL2とします。スクロール・コマンドのパラメータは図 4-5-5のようになっていますから、これを順番に送ります。こうしてパラメータを連続的に変化させることで、ワード単位の滑らかなスクロールが実現されるわけです。スクロールに関するコマンド／パラメータの送信手順は次のとおりです。

◆ スクロールコマンドをポートA0Hへ送る

スクロールコマンド＝70H

0	1	1	1	←　RA　→

(注) コマンドの下位４ビットのRAはGDC内部のRAMアドレスを示し、第１パラメータを格納するアドレス（RA＝0）を与えます。このアドレスは、パラメータ入力毎に自動的にインクリメント（＋１）されます。

◆ スクロールコマンドに対するパラメータを次の順でポートA2Hへ送る

①SAD1L＝AREA1トップアドレスの下位バイト

②SAD1H＝AREA1トップアドレスの上位バイト

③SL1L　＝AREA1のライン数の下位４ビットを上位４ビットとした値
(b3〜b0＝0000)

④SL1H　＝AREA1のライン数の上位６ビットを下位４ビットとした値
b7：1によるインクリメントを選択（DAD＋2＝0）
b6：IMビット（5MHzで1、2.5MHzで0）

⑤SAD2L＝AREA2トップアドレスの下位バイト

⑥SAD2H＝AREA2トップアドレスの上位バイト

⑦SL2L　＝AREA2のライン数の下位４ビットを上位４ビットとした値
(b3〜b0＝0000)

⑧SL3H　＝AREA2のライン数の上位６ビットを下位４ビットとした値
b7：1によるインクリメントを選択（DAD＋2＝0）
b6：IMビット（5MHzで1、2.5MHzで0）

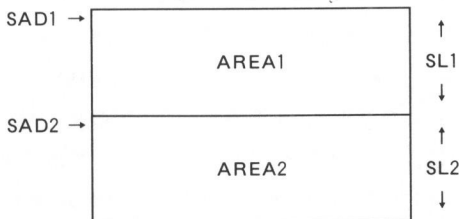



図 4-5-4　表示画面との対応

RA	7	6	5	4	3	2	1	0
0	←			SAD1L				→
1	←			SAD1H				→
2	←	SL1L		→	0	0	0	0
3	*	IM	←		SL1H			→
4	←			SAD2L				→
5	←			SAD2H				→
6	←	SL2L		→	0	0	0	0
7	*	IM	←		SL2H			→

＊:DAD＋2

図 4-5-5　内蔵RAMマップ
RA:GDC内部のRAMアドレス

DAD＋2	機　能
0	"1"によるインクリメント(DAD＋1→DAD)
1	"2"によるインクリメント(DAD＋2→DAD)

DAD:Display Address

IM	表 示 制 御	L／R 制 御
0	2クロックに1回表示アドレスをインクリメント	CSRFORMコマンドの規定値使用
1	4クロックに1回表示アドレスをインクリメント	L／R＝0に強制

IM:Image

ところで、PC-9801シリーズも E／F、VM、VX、RA、……と機能がアップしてきました。GDC に関しても、現在では動作クロックに5MHzと2.5MHzの2通りがあり、ユーザーが選択して使用できるようになっています。5MHzモードであれば、単純計算で2.5MHzの2倍の描写速度が得られることになりますが、それぞれのモードはディップスイッチ2番の8によって選択します(OFFで2.5MHz、ONで5MHz)。変更に際してはシステムを再起動しなければなりません。

なお、クロック数を切り換えた場合、さらに GDCへのパラメータもクロックに対応させる必要があります。スクロール・コマンドの場合、2.5MHzでIMビット＝0、5MHzでIMビット＝1となります。もちろん、ここでのプログラム (**LIST 4-2**) もディップ・スイッチの状態によって両者を区別するようにしてあります。また、MS-DOSのグラフィックドライバを使ったグラフィックスシステムの初期化作業では、GDCの2.5MHzをサポートしていないために、この場合に限ってROM内ルーチンを利用しています。では (**LIST 4-2**) を実行してみましょう。

LIST 4-2

```
;***********************************
;               LIST4-2            *
;***********************************

EXTRN   GSTAT:NEAR,GMD16:NEAR,GTERM:NEAR

CODE    SEGMENT PUBLIC

        ASSUME  CS:CODE,DS:CODE
;
PRINT   MACRO   STRING                          ;;文字列出力用マクロ定義
        MOV     DX,OFFSET STRING
        MOV     AH,09
        INT     21H
        ENDM
;
GDCOUT  MACRO   REG                             ;;GDCへのパラメータ出力用マクロ定義
        IFIDN   <REG>,<AH>
          XCHG    AL,AH
        ENDIF
        OUT     0A0H,AL
        nop
        nop
        ENDM
;
VRAMW   MACRO   VSEG                            ;;G.VRAMへのWRITE
        LOCAL   NOTW
        MOV     AX,VSEG
        MOV     ES,AX
        XOR     AX,AX
        SHR     DX,1
        JNB     NOTW
        MOV     AX,0AAAAH
NOTW:   MOV     CX,25*40
        PUSH    DI
        REP     STOSW
        POP     DI
        ENDM
;
GETKEY  MACRO                                   ;;1文字入力
        MOV     DL,0FFH
        MOV     AH,6
        INT     21H
        ENDM
;                                               ; [ESC] キーのアスキーコード
ESCKY   EQU     1BH
;
PMAIN:  CLD
        CALL    GSTAT                           ;グラフィックシステム・スタート
        JB      TEXIT                           ;異常終了であればTEXITへ
        IN      AL,31H                          ;AL←SW2
        TEST    AL,80H                          ;GDCのクロック・チェック
        JE      HZ5M                            ;5MHzならHZ5Mへ
        MOV     AX,4200H                        ;グラフィック画面モード設定コマンドをセットする
        MOV     CX,0C000H                       ;640×400ドット8色モード設定
        INT     18H                             ;ROM内ルーチン・コール
        MOV     GDCMD,0                         ;2.5MHzとする
        JMP     HZ25                            ;HZ25へジャンプ
HZ5M:   CALL    GMD16                           ;640×400ドット4096色モード設定
        JB      TEXIT                           ;異常終了であればTEXITへ
```

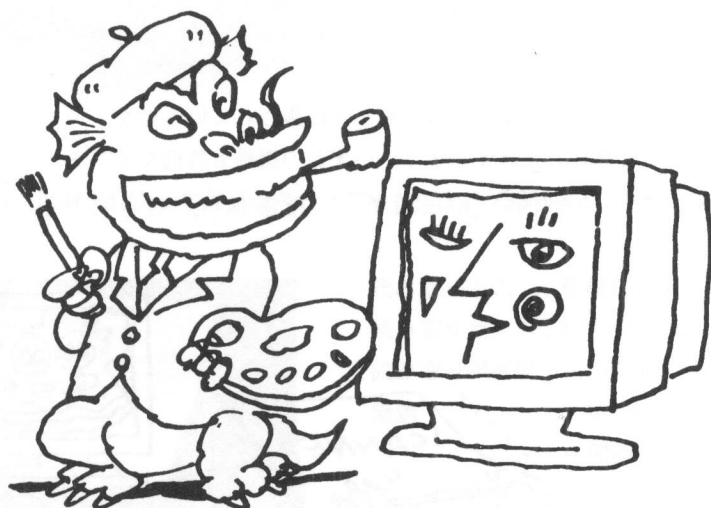
```
HZ25:   PRINT   TXCLR               ;テキスト画面クリア
        MOV     CX,16               ;
        MOV     DI,0                ;
TLP2:   PUSH    CX                  ;
        DEC     CX                  ;
        CALL    HAIKEI              ;  背景の描画
        ADD     DI,25*80            ;
        POP     CX                  ;
        LOOP    TLP2                ;
STRT1:  CALL    SCROLL              ;スクロール
        GETKEY                      ;1文字入力
        CMP     AL,ESCKY            ;  [ESC]キーが押されていなければSTRT1へ
        JNE     STRT1               ;
        CALL    GTERM               ;グラフィックシステムの終了
        PRINT   TKEEP               ;テキスト文字の属性を元に戻す
TEXIT:  MOV     AX,0C00H            ;  キーボード・バッファ・クリア
        INT     21H                 ;
        MOV     AX,4C00H            ;  MS-DOSシステムへ戻る
        INT     21H                 ;
;
SCLDOT  EQU     2                   ;SCrolL DOT number
SCLPTC  EQU     SCLDOT*40           ;SCrolL PiTCh
SCLAD1  DW      4000H+SCLPTC        ;SCrolL ADdress 1
SCLNO1  DW      0                   ;SCrolL NO 1
SCLNO2  DW      0                   ;SCrolL NO 2
GDCMD   DB      40H                 ;GDCモード
;
SCROLL  PROC
        CALL    VWAIT               ;ウェイト
FIFOCK: IN      AL,0A0H             ;
        TEST    AL,4H               ;  GDCバッファの空きチェック
        JZ      FIFOCK              ;
        MOV     AL,70H              ;  スクロール・コマンドの送出
        OUT     0A2H,AL             ;
        MOV     SI,SCLAD1           ;
        SUB     SI,SCLPTC           ;
        MOV     SCLAD1,SI           ;
        MOV     CX,SCLDOT           ;  各パラメータの更新
        CMP     SI,4000H            ;
        JG      CALDAT              ;
        XOR     AX,AX               ;
        MOV     SCLNO1,AX           ;
        MOV     DI,4000H+3E80H      ;
        MOV     SCLAD1,DI           ;
        SUB     DI,SCLPTC           ;
        MOV     BX,190H             ;
        MOV     SCLNO2,BX           ;
        SUB     BX,CX               ;
        MOV     DX,CX               ;
        JMP     START               ;
CALDAT: MOV     BX,SCLNO1           ;
        ADD     BX,CX               ;
        MOV     SCLNO1,BX           ;
        MOV     DX,SCLNO2           ;
        SUB     DX,CX               ;
        MOV     SCLNO2,DX           ;
        MOV     DI,4000H            ;
START:  MOV     CX,4                ;
        MOV     AX,SI               ;
        GDCOUT  AL                  ;
        GDCOUT  AH                  ;
        MOV     AX,BX               ;  GDCへパラメータを送出
        SHL     AX,CL               ;
```

```
        GDCOUT  AL                      ;
        OR      AH,GDCMD                ;
        GDCOUT  AH                      ;
        MOV     AX,DI                   ;
        GDCOUT  AL                      ;
        GDCOUT  AH                      ;
        MOV     AX,DX                   ;
        SHL     AX,CL                   ;
        GDCOUT  AL                      ;
        OR      AH,GDCMD                ;
        GDCOUT  AH
        RET
SCROLL  ENDP
;
VWAIT   PROC
VWAIT1: IN      AL,0A0H                 ;
        TEST    AL,20H                  ;
        JNE     VWAIT1                  ;  ウェイト
VWAT2:  IN      AL,0A0H                 ;
        TEST    AL,00100000B            ;
        JE      VWAT2                   ;
        RET
VWAIT   ENDP
;
HAIKEI  PROC
        MOV     DX,CX                   ;
        VRAMW   BLUE                    ;
        VRAMW   RED                     ;
        VRAMW   GREEN                   ;  背景の描画
        MOV     CL,GDCMD                ;
        JCXZ    HAIRT                   ;
        VRAMW   ITSTY                   ;
HAIRT:  RET
HAIKEI  ENDP
;
TXCLR   LABEL BYTE                      ;テキスト・クリア  &  文字の属性の保存
        DB      1BH,"[2J"
        DB      1BH,"[s",1BH,"[>5h",1BH,"[1>h$"

TKEEP   LABEL BYTE                      ;文字の属性の復元
        DB      1BH,"[u",1BH,"[>5l",1BH,"[1>l$"

CODE    ENDS

STACK   SEGMENT STACK                   ;スタック・セグメントの定義
        DW      100H    DUP(?)
STACK   ENDS

BLUE    SEGMENT AT 0A800H               ;B面のセグメントの定義
BLUE    ENDS

RED     SEGMENT AT 0B000H               ;R面のセグメントの定義
RED     ENDS

GREEN   SEGMENT AT 0B800H               ;G面のセグメントの定義
GREEN   ENDS

ITSTY   SEGMENT AT 0E000H               ;I 面のセグメントの定義
ITSTY   ENDS

        END
```

第5章　魅惑の大型グラフィックス

いわゆるCG（コンピュータ・グラフィックス）とアクション・ゲームで使われるパターン画との境界線はどこにあるのでしょうか。どちらも同じグラフィック画面に描かれた絵であり、違いを明確に規定するものはありません。

とはいえ、CGはデータをメモリに保存し切れないほど大きい絵……というばく然とした認識はあるようで、一昔前にはCGといえば BASICのLINE,PAINT,PSET命令によって描かれるのが普通でした。これが徐々にマシン語化され、高速ラインルーチンとか高速ペイントルーチンなどが開発されたのです。このままの状態が続けば、パターン画はG.VRAMに直接データを入れるもの、CGはライン／ペイントで描くものという定義ができていたかもしれません。

……が、それがすでに過去の話であることは明らかです。いまでは、ライン／ペイントでは作画不可能なドット絵も登場し、しかも当然のように高速に表示されています。そこにどのような進歩があったのか、過去をちょっぴり気にしながら身近になったCGをエンジョイしちゃいましょう。

5－1　スキャナによる画像入力

実際の表示方法がライン／ペイントであれ何であれ、大きな絵を画面に描く前にはそれなりに原画（下絵）を準備するのが普通です。いきなり真っ黒な画面に向かうのは、どんなに優秀なグラフィック・ツールを使おうと、全体の構図がつかみにくく作業効率は低下します。したがって、原画を描くということは昔も今も変わらぬCGの基礎工程といえるでしょう。

次に、それに基づいて画面上にアウトラインを描写するわけですが、ハードウェア的なサポートがないころは、透明ラップに原画を写してディスプレイに貼り付け、それをなぞるようにデジタイズしたものです。非常に原始的な方法ですが、これでも方眼紙に描いた絵から座標を求めるよりは楽だったのです。

その後、色々な画像入力装置（デジタイザーなど）が開発されましたが、いま最も注目を浴びているのがイメージスキャナです。これは、写真や絵をコピー機のようなもので読み取り、それを紙ではなく画面上に複写するものです。発売当初はかなり高価な商品だったため、個人で所有するには相当勇気が要りましたが、最近は白黒専用の普及品から本格的なカラースキャナまで種類も豊富に各社から発売されるようになりました。

こういった状況から、ゲームで使用されるCGには【原画】→【スキャナ取り込み】→【ドット修正／着色】という作画工程が確立され、デジタイズの苦労は大幅に軽減されたのです。本書でも、この工程を前提にCGの解説をしていきますが、本章最後に紹介するグラフィック・ツール単独でも従来方式の作画は可能です。ただし、スキャナが1枚の絵を数十秒で取り込むのに対し、数時間の労力を要することを覚悟してください。

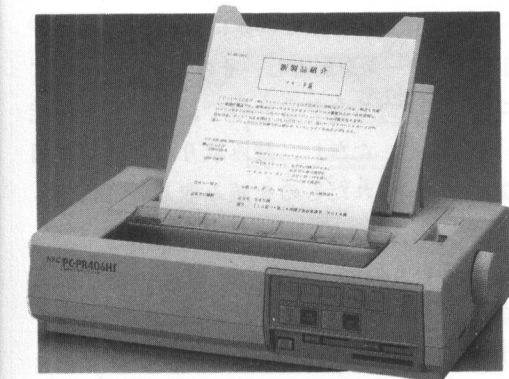
では、現在どのようなスキャナが市販されているのか、いくつかの製品を見ながら特徴を比較してみましょう。いずれの機種も特別なインターフェースボードは必要とせず、RS-232C のインターフェースに接続するだけで使用できます。

◆ PC-IN505 ◆　ＮＥＣ（日本電気）　¥138,000（税別）

読み取り方式　：原稿固定型平面操作方式
読み取りサイズ：A6判（最大105x160mm）
読み取り線密度：90～400 DPI（10DPIステップ）
210DPI以上は200DPIのデータを
拡大処理
階調　　　　　：２値、疑似中間調（ディザ1,2,3）
中間調（2,4,8,16,32,64階調）
カラー読み取り：RGB 面順次
外形　　　　　：340(W)×220(D)×100(H)mm

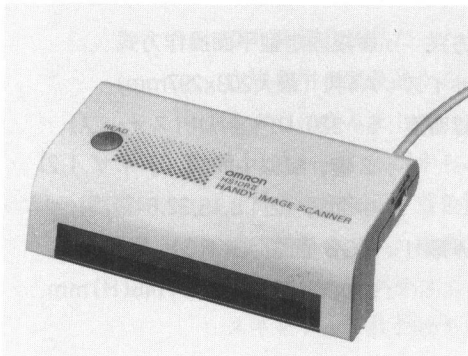
◆ PC-IN506 ◆　ＮＥＣ（日本電気）　¥228,000（税別）

読み取り方式　：原稿固定型平面操作方式
読み取りサイズ：A4判（最大203x297mm）
読み取り線密度：5～320 DPI（1DPIステップ）
階調　　　　　：２値、疑似中間調（ディザ 1,2）
中間調（2,4,8,16,32,64階調）
カラー読み取り：RGB 面順次／RGB 線順次
外形　　　　　：500(W)×360(D)×146(H)mm

◆ PC-PR406HS ◆　ＮＥＣ（日本電気）　¥138,000（税別）

読み取り方式　：原稿移動型シリアル走査方式
読み取りサイズ：B4判（最大205x297mm）
読み取り線密度：90,160,180 DPI
階調　　　　　：２値、疑似中間調（ディザ）
中間調（4,8,16階調）
プリンタ部　　：日本語カラー熱転写
外形　　　　　：410(W)×246(D)×103(H)mm

◆ HS7RⅡ ◆　　オムロン　　　　　　　　　¥39,800（税別）

読み取り方式　：スキャナ手動走査方式
読み取りサイズ：A6幅（最大105mm）
読み取り線密度：90,120,160,180,200,240,300,400 DPI
200DPI以外は200DPIのデータを縮小／拡大処理
階調　　　　　：2値、疑似中間調（ディザ）
外形　　　　　：146(W)×111(D)×40(H)mm

◆ HS10RⅡ ◆　　オムロン　　　　　　　　　¥49,800（税別）

読み取り方式　：スキャナ手動走査方式
読み取りサイズ：A6幅（最大104mm）
読み取り線密度：90,120,160,180,200,240,300,400 DPI
200DPI以外は200DPIのデータを縮小／拡大処理
階調　　　　　：2値、疑似中間調(ディザ／誤差拡散)
外形　　　　　：130(W)×70(D)×41(H)mm

コピー機型、プリンター一体型、ハンディ型と、本体タイプの違いがそれぞれ読み取り方式の違いになっています。コピー機型以外は、読み取り位置を一定させることが難しいので、同じ絵を階調を変えて読み取ろうとした場合など、ある程度のズレが生じるのは避けられません。ユニークなのはプリンタと一体になっているPC-PR406HSで、ハガキサイズのものなら読み取りデータを記憶し、そのままプリントアウトすることができます。これはパソコンをまったく介さないので、スキャナ機能を利用した簡易コピー機といってもいいでしょう。ただし、読み取りはプリンタ用紙を送るように行いますから、本や箱などに印刷されたものを読み取ることはできません。

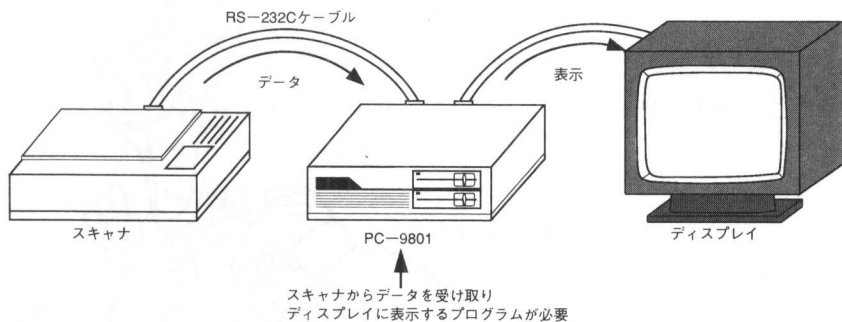
読み取り線密度の DPI (Dot Per Inch) というのは、1インチ（約25mm）を何ドットに分解して読み取るかという単位です。例えば、200DPIは1インチを 200ドットに分解することですから、1ミリ当りに換算すると約8ドットです。グラフィック画面は横 640ドットですから、読み取った通りに表示すれば画面幅いっぱい（640ドット）が約8cmに相当するわけです。現在普及しているディスプレイは14イン

チ（対角線長）のものが主流ですから、感覚的には8cmが24〜25cmに拡大表示されたことになります。したがって、原画のサイズイメージを生かすならば、70DPI前後が最適ということになりますが、原画を描く前にサイズ合わせをして感覚を把握しておくといいでしょう。

階調の2値とは、原画を完全に白／黒で識別するもので、中間色（灰色など）は濃度によって白か黒のどちらかになります。これをカバーするのが疑似中間調で、ディザ法によって濃淡を表現します。ディザ法については、各製品のマニュアルに詳しい説明がなされていますから、ここでは省きます。要するに、ドット密度によって濃淡を表現する方法のことで、新聞の白黒写真を虫メガネで拡大したようなものです。

一方、中間調識別機能があるものは、濃淡をレベルで識別することができます。例えば16階調の場合であれば、白〜灰色（14レベル）〜黒と16レベルに分けて読み取ることができるわけです。ただし、PC-9801 の場合は最大で16色しか表現できませんから、実際にはこの機能をカラーで使いこなすことはできません。

なお、カラー読み取りのできるPC-IN505／PC-IN506には RGB面順次と RGB線順次という区別があります。これはカラーをプレーン順（G→R→B）に3回読んでいくか、1ラインずつ3回読んでいくかという違いです。構造的にキャリッジ（読み取り部）が3往復する面順次方式では、微妙にドットカラーがズレる危険性がありますが、線順次なら確実にカラーを捕らえることができます。ただし、640×400ドット程度ではその心配はないと考えていいでしょう。現在は価格的なことから面順次方式が普及していますが、将来は線順次方式が主流になるといわれています。

さて、スキャナの種類やだいたいの特徴は理解できたと思いますが、実際にスキャナを使うためにはスキャナからのデータを受け取るプログラムが必要です。つまり、RS-232C インターフェースを通じて送られてきたデータを受け取り、それをG.VRAMに入れるというプログラムです。

PC-IN505／506には PC-9801と PC-88VA用のソフトウェアがディスクで提供されていますが、それ以外の製品にはソフトウェアのサポートがありません。一応プログラムを組むための情報が各マニュアルに載っていますが、マシン語に相当精通し

ていないと自分でプログラムを組むのは難しいかもしれません。というのは、一口にデータの受け取りといっても、実体はスキャナという外部機器をコントロールすることだからです。スキャナの仕様を理解するだけでなく、RS-232C を通してコマンドやデータを送受信できるだけの知識も要求されています。それに、ゲームに直接関与しないことを、ここで長々と解説するだけのゆとりもありません。

それならば、そういった細かなことを解説するより、使いやすいツールにして提供したほうがいい……ということで、本書ではすべての製品に使用できるPC-9801専用のソフトウェア『SCAN98』を用意しました。詳しい操作方法は最後の節で解説しますが、単にデータをG.VRAMに入れるだけでなく、次のような特殊機能が追加されています。

① 表示方向の縦/横指定

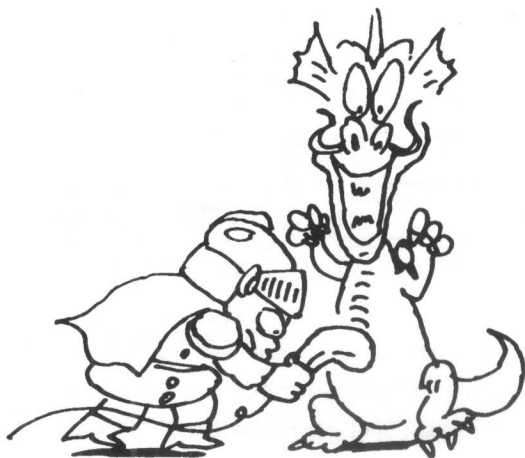
　スキャナで読み取ったデータを、縦方向（上から下）でも横方向（左から右）でも表示させることができます。この機能により、読み取り幅の小さいスキャナでも、横長の原画を読み取れることになります。

② 1/2 縮小表示

　14インチディスプレイの場合、90DPI が最小では原画を拡大したイメージでしか表示できません。1/2 縮小表示機能を利用することにより、表示サイズを細かく設定できるようになります。

③ カラースキャナへの対応

　モノクロ、カラー（面順次、線順次）いずれにも対応できるようにしてあります。

こうして取り込まれた画像は、即座にグラフィック・ツール『PMAN98』でエディットできるよう環境を整えてあります。もちろん、作成した絵は保存／再現が自由自在です。その秘密は……。次の節に書いてあります。

5－2　グラフィック・データの圧縮と展開

オモチャ箱をひっくり返したような乱雑な部屋でも、その気になって整理をすれば案外すっきりとまとまるものです。では、ここでいう「整理」とは、いったい何をどうすることなのか……なんて難しく考えながら片付ける人はいないでしょう。たいていは無意識のうちに同じ種類のものをまとめているはずです。

しかし、大人と子供では整理能力が違うように、バラバラの中から共通項を発見するというのは意外と知的作業なのです。例えば、本箱に本をしまうにしても、幼児はただ並べるだけですが、やがて種類別に分けることを覚えます。もう少し大きくなると、発行月順に並べたり、背の順に並べたり、あるいはよく読む順に並べたり……と、かなり個人差が現れてきます。こうなると、一概にどれが最善の整理法ということはできません。

つまり、共通項というものは必ずしも一定したものではなく、アイデアと状況により変化するものなのです。グラフィック画面をフルに使えば、80（横）×400（縦）×4（プレーン）＝128000バイトという大量のデータがG.VRAMに納まっていることになります。このままデータ化すれば、セグメント2つ分がまるまるデータ領域になってしまいます。そこで、共通項を捜してデータをまとめ、表示する時に元の姿に展開しようというのです。これがデータ圧縮／展開の基本です。

問題は、本の並べ方の例でもわかるように、すべてに最適となる圧縮法はないということです。ある絵に対しては強力な圧縮がかかるのに、別の絵に対しては圧縮どころか逆にデータが増えてしまうということだってあるのです。次のデータ圧縮法は、連続した同じ数値に対し圧縮をかけるという、最も基本的な圧縮サンプルです。

圧縮前のデータ：55 55 55 55 55 12 FF FF FF FF
圧縮後のデータ：55 55 03 12 FF FF 02

ここでの圧縮サインは同じ数値の連続です。同じ数値が2つ続いたら、次の数値でその値があといくつ連続するかを示しています。この圧縮法により、この例では10個のデータが7個になっています……が。

圧縮前のデータ：55 55 FF FF 55 55 FF FF 55 55
圧縮後のデータ：55 55 00 FF FF 00 55 55 00 FF FF 00 55 55 00

今度は、圧縮どころかデータが増えてしまいました。圧縮には、こういった危険性が常に潜んでいるのです。ご存じの方もいるかもしれませんが、以前ある雑誌で『グラフィック・データ圧縮コンテスト』というものを企画したことがあります。それは3枚の課題絵を提供して圧縮率を競い合うというものですが、コンテストです

から圧縮法は応募者によってすべて違います。その結果、絵によって順位はバラバラになることが事実として確認されたのです。これは、ある程度予測していたことでしたが、これほどハッキリと絵の影響を受けるとは思っていませんでした。

しかし、「だから汎用の圧縮法は存在しない」というのではありません。この事実が示したことは、「総合して優秀なものが最も汎用性の高い圧縮ルーチンである」ということの裏返しなのです。これから紹介する圧縮法は、私が全力を投入して開発したバージョンを一部（かなり）カットしたものですが、それでも十分に実用に耐えるものです。カットした理由は、ページ数の問題が最大のものですが、さらに独自のアイデアを追加してほしいという希望もあったからです。それがなければ圧縮ルーチンをソースリストで紹介する意味がなくなってしまうでしょう。

なお、ここでの圧縮は数ある圧縮法の中の一例であり、すべての圧縮が似たような考え方に基づいているというわけではありません。例えば、スキャナを購入するとマニュアルには多段階圧縮法というビット単位の圧縮の説明がありますが、データをまったく違う視点からまとめています。ただ、多段階圧縮はカラーグラフィックスに適した圧縮ではないため、そのままゲーム等に応用するのは有効とはいえません。とはいえ、この考え方を基本に独自の工夫改良を加えてゲーム用の圧縮とした例もありますから、興味のある方はスキャナのマニュアルをよく読んで参考にするといいでしょう。

では、本書の圧縮法で使われている圧縮データの構造を示します。データは、G.VRAMの内容を1バイトずつ縦方向に読んでいますが、これは画面が絵であることから横方向より縦方向に共通性が高いからです。よくわからない場合は、図 5-2-1からどちらのペン先が絵を描きやすいか考えてください。

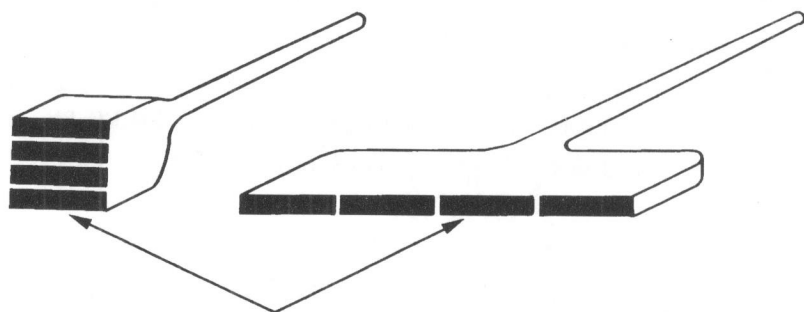



図 5-2-1

一般的な圧縮の基本は前後のデータとの共通性ですが、グラフィック画面の場合は前後のデータだけでなく、他のプレーンのデータも比較の対象となります。実は、これがグラフィック・データ圧縮の最大のポイントで、単に他のプレーンと同じかどうかを調べるだけでもかなりの効果があります。さらに、他のプレーンデータをANDや XORを取ってから調べると、そこには複数の共通性が存在していることもあり、どれを選択するかはまさに思考ゲームといっても過言ではありません。ここ

では、そういったすべての可能性を追求してはいませんし、複数の共通性の中から必ずしも最良の選択がなされているとも限りません。この先、どこまでアイデアと効率を追求できるか、グラフィック・データの圧縮はここからが真のスタートといえるでしょう。もちろん、私自身も新たなアイデアとアルゴリズムによる構想は完了しています。あとは、それをプログラミングするきっかけが必要なだけです。そのきっかけとは……、読者の皆さんによって私の圧縮法が過去形になることです。オリジナルな圧縮法の発見／開発の朗報をお待ちしています。

それでは、本プログラムでの圧縮データの構造と、圧縮のサインを紹介しましょう。

圧縮データの構造

圧縮データは、セグメントDATSEG1のオフセットアドレスDTOPから、最大でDTMAXだけ生成され、それを越えるデータは無視されます。

DTTOP　　：プレーン別にデータの有無、A／B圧縮の区別をする

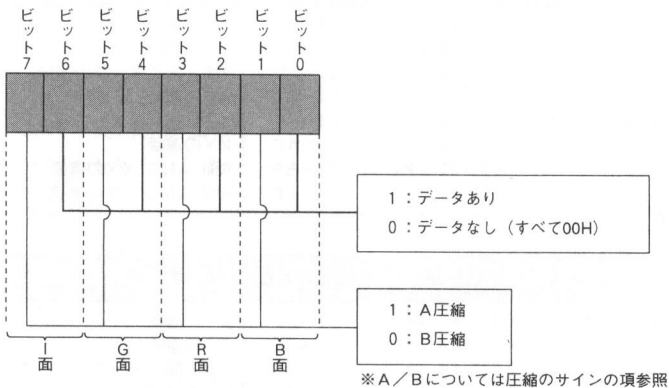



図 5-2-2

DTTOP+1　：横バイト数

DTTOP+2　：縦ライン数

DTTOP+4　：圧縮データ識別コード（1234H固定）

DTTOP+6　：予備

⋮

DTTOP+53　：予備

DTTOP+54～：圧縮されたデータ

圧縮のサイン

圧縮のサインはデータの上位／下位が同じ数値の場合（A圧縮＝すべて、B圧縮＝00,FF,55,AA,33,66,99,CCのみ）で、その次に連続するデータ数が示されます。ポイントはこの連続数の数え方で、一定の範囲内で数えることによって圧縮の内容に変化をつけています。

例えば、連続数の範囲（□□の部分）が 81～FF となっていれば、ビット 7 が圧

縮の内容を示すフラグで、1～7F がその連続する数というわけです。もし、連続する数が□□の範囲を越える場合は、□□をその範囲での最大値とした上で、次の2バイト（＃＃ ＃＃ の部分）で連続する総数を表します。

X と Yは1バイトの数値の上位あるいは下位4ビットのことで、XX＝上位／下位が同じ数値、XY＝上位／下位が違う数値という意味です。また、データ（XX）がローテートする場合は（※）で示し、55,AA については1ローテート、それ以外の数値は2ローテートします。

★B（ブルー）面 ＆ I（輝度）面

圧縮のサイン	連続数	□□の範囲	内容
00 00 XY	□□（## ##）	04～FF	XYの連続
00	□□（## ##）	01～FF	00の連続
FF	□□（## ##）	01～FF	FFの連続
XX	□□（## ##）	01～7F	XXの連続（※）
XX	□□（## ##）	81～FF	XXの連続

★R（レッド）面

圧縮のサイン	連続数	□□の範囲	内容
00 00 XY	□□（## ##）	04～FF	XYの連続
00	□□（## ##）	01～7F	00の連続
00	□□（## ##）	81～FF	B面と同じデータの連続
FF	□□（## ##）	01～FF	FFの連続
XX	□□（## ##）	01～7F	XXの連続（※）
XX	□□（## ##）	81～AF	XXの連続
XX	□□（## ##）	B1～BF	B面 AND XXの連続
XX	□□（## ##）	C1～FF	B面 AND XXの連続（※）

★G（グリーン）面

圧縮のサイン	連続数	□□の範囲	内容
00 00 XY	□□（## ##）	04～FF	XYの連続
00	□□（## ##）	01～7F	00の連続
00	□□（## ##）	81～FF	B面と同じデータの連続
FF	□□（## ##）	01～7F	FFの連続
FF	□□（## ##）	81～FF	R面と同じデータの連続
XX	□□（## ##）	01～3F	XXの連続（※）
XX	□□（## ##）	41～5F	XXの連続
XX	□□（## ##）	61～6F	B面 AND XXの連続
XX	□□（## ##）	71～7F	R面 AND XXの連続
XX	□□（## ##）	81～BF	B面 AND XXの連続（※）
XX	□□（## ##）	C1～FF	R面 AND XXの連続（※）

表を見ただけでは圧縮の特徴がわからないかもしれませんが、これは主にグラフィックスのペイントされている部分に共通性を求めたものです。つまり、ゲーム等に使用されている一般的なグラフィックスに対して効果的な圧縮ということです。また、R／G面については他面のデータを参照してデータを作成することができますから、実際にはペイント以外の部分に対しても圧縮をかけていることになります。参考までに圧縮のサンプルを載せておきますので、表と照らし合わせながらデータ構造を確認してください。

圧縮の例

	圧縮前		圧縮後
B面	FF,FF,FF,FF,FF,F0,F0,F0,F0,F0,F0,73	→	FF,05,00,00,F0,06,73
R面	55,AA,55,AA,55,A0,50,A0,50,A0,50,22	→	55,CC
G面	33,33,33,33,33,F0,F0,F0,F0,F0,F0,73	→	33,65,00,87
I面	FF,FF,FF,FF,FF,FF,FF,F0,F0,F0,F0,F0	→	FF,07,00,00,F0,05

この場合、トータルして48のデータが19に圧縮されていますから、約 40%の圧縮率ということになります。ここまで圧縮の特徴を知ると、きっと色々な疑問やアイデアが涌いてくるかもしれません。ただ、圧縮も他のプログラム同様、推定結果と実行結果が違うことがよくあります。いいと思って投入したアイデアが、結果的に役に立たないとわかるのはプログラムを組んでからの話です。疑問やアイデアは机上の空論に終わることなく、ぜひプログラムとして自分の目で結果を確認してください。おそらく、それは単に圧縮理論の向上になるだけでなく、プログラミング技術が飛躍するチャンスとなるでしょう。

なお、ここで作成するプログラム（LIST 5-1）は、オブジェクト・ファイル形式にしてありますから、利用する場合には他のプログラムとリンクさせることになります。このうち、圧縮／展開プログラムは次の節で紹介するグラフィック・ツールにおいて実際に画面のロード／セーブ用としても使用しています。

また、グラフィック・ツール（PMAN98）のオブジェクト・ファイルも格納してありますから、オリジナルな圧縮／展開プログラムを作成した場合でも、次にあげるラベル名さえ同じであれば、リンケージが可能となっています(なお、リンク時には、PMAN98.OBJとLIST5-1.OBJの他にLIST1-0.OBJとLIST2-1.OBJが必要となります)。

＊グラフィックツール『PMAN98』で使用しているラベル名

ラベル名	型	内容
XSIZE	バイト	圧縮の横バイト数
YSIZE	ワード	圧縮の縦バイト数
DTADR	ワード	圧縮データ・エンド・アドレス
PTOP	ワード	圧縮画面の左上アドレス
OVERS	バイト	オーバー・フロー・サイン
DPRES	プロシージャ	圧縮用プロシージャ名
OPEN3	プロシージャ	展開用プロシージャ名
DATSEG1		圧縮データ格納エリアセグメント名（コンバイン・タイプにPUBLICを指定する）
DTOP	バイト	圧縮データ格納エリアの先頭オフセット・アドレス

LIST 5-1

```
;***********************************
;*              LIST5-1            *
;***********************************

PUBLIC  XSIZE,YSIZE,DTADR,PTOP,DPRES,OPEN3,DTOP,OVERS

CODE    SEGMENT PUBLIC

        ASSUME  CS:CODE,DS:DATSEG1

PTOP    DW      0                       ;圧縮画面の左上アドレス
XSIZE   DB      80                      ;圧縮の横バイト数
YSIZE   DW      400                     ;圧縮の縦ライン数
DTADR   DW      0                       ;圧縮データエンドアドレス
DDTOP   DW      OFFSET DTOP             ;展開／表示データ先頭アドレス
XYPOS   DW      0                       ;展開／表示の左上アドレス
ATOP    DW      0                       ;A圧縮のデータエンド
BTOP    DW      0                       ;B圧縮のデータエンド
DCONT   DB      0                       ;データ圧縮中のサイン
OVERS   DB      0                       ;オーバーフローサイン

BLUE    EQU     0A800H                  ;B面セグメントアドレス
RED     EQU     0B000H                  ;R面セグメントアドレス
GREEN   EQU     0B800H                  ;G面セグメントアドレス
ITSTY   EQU     0E000H                  ;I面セグメントアドレス
DTMAX   EQU     0FE00H                  ;圧縮エリアの最大値
DSTART  EQU     6                       ;圧縮データスタート値

DPRES:  ASSUME  DS:DATSEG1
        MOV     AX,DATSEG1              ;AX=DATSEG1
        MOV     DS,AX                   ;DS=AX=DATSEG1:圧縮データ用セグメント
        MOV     AL,11111111B            ;
        MOV     DTOP,AL                 ;
        MOV     AL,CS:XSIZE             ; データエリア初期化
        MOV     DTOP+1,AL               ;
        MOV     AX,CS:YSIZE             ;
        MOV     WORD PTR DTOP+2,AX      ;
        MOV     AX,1234H                ;識別コード:1234Hであれば圧縮データとする
        MOV     WORD PTR DTOP+4,AX      ;識別コードを格納する
        MOV     BX,OFFSET DTOP+DSTART   ;
        MOV     CS:DTADR,BX             ;データエンドアドレス初期化
        MOV     CS:OVERS,0              ;オーバーフローサイン初期化
        MOV     BX,OFFSET BPRES         ;
        MOV     DX,OFFSET TOBPR         ; B面を圧縮
        MOV     AX,BLUE                 ;
        CALL    PDATA                   ;
        MOV     BX,OFFSET RPRES         ;
        MOV     DX,OFFSET TORPR         ; R面を圧縮
        MOV     AX,RED                  ;
        CALL    PDATA                   ;
        MOV     BX,OFFSET GPRES         ;
        MOV     DX,OFFSET TOGPR         ; G面を圧縮
        MOV     AX,GREEN                ;
        CALL    PDATA                   ;
        MOV     BX,OFFSET BPRES         ;
        MOV     DX,OFFSET TOBPR         ; I面を圧縮
        MOV     AX,ITSTY                ;
        CALL    PDATA                   ;
        RET
```

```
PDATA:  MOV     CS:KPOINT,BX           ;
        MOV     CS:KRETAD,DX           ;  A圧縮を実行し無データ(すべて00H)なら
        MOV     CS:KPLANE,AX           ;  NDATAへ
        CALL    TATEA                  ;
        JNB     NDATA                  ;
        CALL    TATEB                  ;
        MOV     BX,CS:ATOP             ;
        MOV     DX,CS:BTOP             ;  B圧縮を実行しA圧縮と比較していい方
        SUB     BX,DX                  ;  を採用する
        JNB     $+5                    ;
        CALL    TATEA                  ;
        MOV     CS:DTADR,DX            ;
        RET                            ;

NDATA:  MOV     BX,OFFSET DTOP         ;
        MOV     AX,CS:KPLANE           ;
        CMP     AX,GREEN               ;
        JE      RBIT4                  ;
        CMP     AX,RED                 ;
        JE      RBIT2                  ;
        CMP     AX,ITSTY               ;  無データの場合に先頭データのフラグで示す
        JE      RBIT6                  ;
        AND     BYTE PTR [BX],0FEH     ;
        RET                            ;
RBIT2:  AND     BYTE PTR [BX],0FBH     ;
        RET                            ;
RBIT4:  AND     BYTE PTR [BX],0EFH     ;
        RET                            ;
RBIT6:  AND     BYTE PTR [BX],0BFH     ;
        RET                            ;

ABCHK:  OR      AL,AL
        JNE     $+3                    ;
        RET                            ;
ABCK2:  CMP     AL,0FFH                ;
        JNE     $+3                    ;
        RET                            ;
CHKUD:  CMP     AL,55H                 ;
        JNE     $+3                    ;
        RET                            ;
        CMP     AL,0AAH                ;
        JNE     $+3                    ;
        RET                            ;
CKUD2:  CMP     AL,33H                 ;
        JNE     $+3                    ;
        RET                            ;
        CMP     AL,66H                 ;
        JNE     $+3                    ;  A／B圧縮について上位／下位が同じか
        RET                            ;  否かを調べる
        CMP     AL,99H                 ;
        JNE     $+3                    ;
        RET                            ;
        CMP     AL,0CCH                ;
        JNE     $+3                    ;
        RET                            ;
UDPON   DW      0C8D0H                 ;
        ROR     AL,1                   ;
        ROR     AL,1                   ;
        ROR     AL,1                   ;
        CMP     AL,DL                  ;
        MOV     AL,DL                  ;
        RET                            ;
```

```
TATEA:  MOV     AX,20FH                 ;
        MOV     CS:SBIT11,AX            ;
        MOV     AX,80FH                 ;
        MOV     CS:SBIT31,AX            ;
        MOV     AX,200FH                ;  A圧縮のための初期化
        MOV     CS:SBIT51,AX            ;
        MOV     AX,800FH                ;
        MOV     CS:SBIT71,AX            ;
        MOV     BX,OFFSET ATOP          ;
        MOV     DX,OFFSET TYPRA         ;
        MOV     CX,OFFSET TYPGA         ;
        MOV     AX,0C8D0H               ;
        JMP     TATE                    ;

TATEB:  MOV     AX,0FD27H               ;
        MOV     CS:SBIT11,AX            ;
        MOV     AX,0F727H               ;
        MOV     CS:SBIT31,AX            ;  B圧縮のための初期化
        MOV     AX,0DF27H               ;
        MOV     CS:SBIT51,AX            ;
        MOV     AX,07F27H               ;
        MOV     CS:SBIT71,AX            ;
        MOV     BX,OFFSET BTOP          ;
        MOV     DX,OFFSET TYPRB         ;
        MOV     CX,OFFSET TYPGB         ;
        MOV     AX,0C3C3H

TATE:   MOV     CS:UDPON,AX             ;
        MOV     CS:KTATE4,BX            ;
        MOV     CS:KEXIST,BX            ;
        MOV     CS:KTYPER,DX            ;  プログラム／ワークエリア初期化
        MOV     CS:KTYPEG,CX            ;
        MOV     BX,CS:DTADR             ;
        MOV     BP,CS:KTATE4            ;
        MOV     CS:[BP+0],BX            ;
        XOR     AL,AL                   ;
        MOV     CS:DCONT,AL             ;
        MOV     CS:KDTCK0,AL            ;
        MOV     BX,OFFSET DTOP          ;
        MOV     AX,CS:KPLANE            ;
        MOV     ES,AX                   ;
        CMP     AX,RED                  ;
        JE      SBIT3                   ;
        CMP     AX,GREEN                ;
        JE      SBIT5                   ;
        CMP     AX,ITSTY                ;
        JE      SBIT7                   ;
SBIT1:                                  ;
        DB      80H                     ;
SBIT11  DW      020FH                   ;
        JMP     TPRESS                  ;
SBIT3:                                  ;  先頭データにA／Bのフラグをセット
        DB      80H                     ;
SBIT31  DW      080FH                   ;
        JMP     TPRESS                  ;
SBIT5:                                  ;
        DB      80H                     ;
SBIT51  DW      200FH                   ;
        JMP     TPRESS                  ;
SBIT7:                                  ;
        DB      80H                     ;
SBIT71  DW      800FH                   ;
```

```
TPRESS: CALL    KPREG               ;
        MOV     BX,CS:PTOP          ;
        MOV     DX,80               ;
        XOR     AL,AL               ; データ圧縮開始
        SBB     BX,DX               ;
        MOV     CL,AL               ;
        MOV     DI,CS:YSIZE         ;
        INC     DI                  ;
        CALL    KPREG               ;
        MOV     BX,CS:DTADR         ;
        MOV     CX,0                ;
        MOV     DH,0                ;
        JMP     CS:KPOINT

BNORM:  MOV     CH,0                ;
BNORM2: MOV     [BX],DL             ;
        INC     BX                  ; 通常のデータ処理(B面)
        CMP     BX,DTMAX            ;
        JB      $+5                 ;
        JMP     MOVER               ;
        MOV     DH,DL
BPRES:  CALL    TDATA               ;
        MOV     DL,AL               ;
BPRE4:  OR      AL,AL               ;
        JE      BPR0                ; データの上位／下位が共通の場合
        CMP     AL,0FFH             ; それぞれのルーチンへ
        JE      BPR0                ;
        CMP     AL,55H              ;
        JE      BPR1                ;
        CMP     AL,0AAH             ;
        JE      BPR1                ;
        CALL    CKUD2               ;
        JE      BPR2
        CMP     AL,DH               ;前回とデータが違えばBNORMへ
        JNE     BNORM               ;
        INC     CH                  ;
        MOV     AL,CH               ; 同じデータの連続が3回以下なら
        CMP     AL,3                ; BNORM＋2へ
        JNE     BNORM2              ;
SAMST:  PUSH    BX                  ;
        MOV     BYTE PTR [BX],4     ;
        DEC     BX                  ;
        MOV     [BX],DL             ;
        DEC     BX                  ;
        MOV     BYTE PTR [BX],0     ;
        DEC     BX                  ; 同じデータが4回以上連続した場合
        MOV     BYTE PTR [BX],0     ; の処理
        POP     BX                  ;
        MOV     CX,0                ;
        CALL    DCON1               ;
SAMSL:  CALL    TDATA               ;
        CMP     AL,DL               ;
        JE      SAMSL2              ;
        JMP     CS:KRETAD           ;
SAMSL2: CALL    INC1F               ;
        JMP     SAMSL

DCON1:  MOV     AL,1                ;
        MOV     CS:DCONT,AL         ; 連続フラグのセット
        RET                         ;

BPR0:   MOV     [BX],AL             ;
        INC     BX                  ;
```

```
        MOV     BYTE PTR [BX],1         ;} 00HまたはFFHが連続した場合の処理
        CALL    DCON1                   ;
BSAML:  CALL    TDATA                   ;
        CMP     AL,DL                   ;
        JNE     TOBPR                   ;
        CALL    INC1F                   ;
        JMP     BSAML                   ;

TOBPR:  MOV     DL,AL                   ;
TOBPR1: INC     BX                      ;
        XOR     AL,AL                   ;} 圧縮処理の終了
        MOV     CS:DCONT,AL             ;
        MOV     CH,AL                   ;
        MOV     DH,AL                   ;
        MOV     AL,DL                   ;
        JMP     BPRE4

BPR1:   MOV     AX,9090H                ;55HまたはAAHが連続した場合の処理
        JMP     BPR22                   ;
BPR2:   MOV     AX,0C8D0H               ;
BPR22:  MOV     CS:KBSPO,AX             ;
        MOV     [BX],DL                 ;
        INC     BX                      ;
        MOV     BYTE PTR [BX],1         ;} 上位／下位が同じデータで連続した場合
        MOV     DH,DL                   ;  の処理
        CALL    TDAT1                   ;
        MOV     DL,AL                   ;
        JB      BSPO2                   ;
        CMP     AL,DH                   ;
        JNE     BSPO                    ;
        MOV     BYTE PTR [BX],82H       ;
BSPL1:  CALL    TDATA                   ;
        CMP     AL,DH                   ;
        JNE     TOBPR                   ;
        CALL    INC8F                   ;
        JMP     BSPL1                   ;

BSPL2:  CALL    TDATA                   ;
        MOV     DL,AL                   ;
        JB      BSPO2                   ;
BSPO:                                   ;
KBSPO   DW      0C8D0H                  ;
BSPO2:  ROR     AL,1                    ;} 圧縮が次の縦列になった場合の処理
        CMP     AL,DH                   ;
        JNE     TOBPR1                  ;
        MOV     DH,DL                   ;
        CALL    INC17                   ;
        JMP     BSPL2                   ;
RPRES:  CALL    TDATA                   ;
        JB      $+5                     ;
        CALL    ROTER                   ;
        MOV     DL,AL                   ;
RPRE4:  MOV     AL,CL                   ;} CL=00HならRNEWへ
        OR      AL,AL                   ;
        JE      RNEW                    ;
        CMP     AL,0FFH                 ;
        JNE     $+5                     ;} CL=0FFHならCFENTへ、それ以外なら
        JMP     CFENT                   ;  CXENTへ
        JMP     CXENT                   ;

RNEW:   MOV     AL,DL                   ;
        MOV     BP,BLUE                 ;} データがB面と同じならBSET0へ
        MOV     ES,BP                   ;
```

```
        CALL    KPREG                           ;
        CMP     AL,ES:[BX]                      ;
        CALL    KPREG                           ;
        JNE     $+5                             ;
        JMP     BSET0                           ;
        OR      AL,AL                           ;
        JNE     $+5                             ; データ=00HならPRO00へ
        JMP     RPR00                           ;
        MOV     AL,55H                          ;
        MOV     CL,AL                           ;
        CALL    KPREG                           ;
        AND     AL,ES:[BX]                      ;
        CALL    KPREG                           ;
        CMP     AL,DL                           ;
        JNE     $+5                             ;
        JMP     BSETX                           ;
        MOV     AL,0AAH                         ;
        MOV     CL,AL                           ;
        CALL    KPREG                           ;
        AND     AL,ES:[BX]                      ;
        CALL    KPREG                           ;
        CMP     AL,DL                           ;
        JNE     $+5                             ;
        JMP     BSETX                           ;
        MOV     AL,33H                          ;
        MOV     CL,AL                           ;
        CALL    KPREG                           ;
        AND     AL,ES:[BX]                      ;
        CALL    KPREG                           ;
        CMP     AL,DL                           ;
        JNE     $+5                             ; データが「B面 AND CL」ならBSETXへ
        JMP     BSETX                           ;
        MOV     AL,66H                          ;
        MOV     CL,AL                           ;
        CALL    KPREG                           ;
        AND     AL,ES:[BX]                      ;
        CALL    KPREG                           ;
        CMP     AL,DL                           ;
        JNE     $+5                             ;
        JMP     BSETX                           ;
        MOV     AL,99H                          ;
        MOV     CL,AL                           ;
        CALL    KPREG                           ;
        AND     AL,ES:[BX]                      ;
        CALL    KPREG                           ;
        CMP     AL,DL                           ;
        JNE     $+5                             ;
        JMP     BSETX                           ;
        MOV     AL,0CCH                         ;
        MOV     CL,AL                           ;
        CALL    KPREG                           ;
        AND     AL,ES:[BX]                      ;
        CALL    KPREG                           ;
        CMP     AL,DL                           ;
        JNE     $+5                             ;
        JMP     BSETX                           ;
        JMP     CS:KTYPER                       ;

TYPRA:  MOV     AL,22H                          ;
        MOV     CL,AL                           ;
        CALL    KPREG                           ;
        AND     AL,ES:[BX]                      ;
        CALL    KPREG                           ;
```

```
        CMP     AL,DL                    ;
        JNE     $+5                      ;
        JMP     BSETX                    ;
        MOV     AL,77H                   ;
        MOV     CL,AL                    ;
        CALL    KPREG                    ;
        AND     AL,ES:[BX]               ;
        CALL    KPREG                    ;
        CMP     AL,DL                    ;
        JNE     $+5                      ;
        JMP     BSETX                    ;
        MOV     AL,88H                   ;
        MOV     CL,AL                    ;
        CALL    KPREG                    ;
        AND     AL,ES:[BX]               ;
        CALL    KPREG                    ;
        CMP     AL,DL                    ;
        JNE     $+5                      ;
        JMP     BSETX                    ;
        MOV     AL,0DDH                  ;
        MOV     CL,AL                    ;
        CALL    KPREG                    ;
        AND     AL,ES:[BX]               ;
        CALL    KPREG                    ;
        CMP     AL,DL                    ;
        JNE     $+5                      ;
        JMP     BSETX                    ;
        MOV     AL,11H                   ;
        MOV     CL,AL                    ;  データが「B面 AND CL」ならBSETXへ
        CALL    KPREG                    ;
        AND     AL,ES:[BX]               ;
        CALL    KPREG                    ;
        CMP     AL,DL                    ;
        JNE     $+5                      ;
        JMP     BSETX                    ;
        MOV     AL,44H                   ;
        MOV     CL,AL                    ;
        CALL    KPREG                    ;
        AND     AL,ES:[BX]               ;
        CALL    KPREG                    ;
        CMP     AL,DL                    ;
        JNE     $+5                      ;
        JMP     BSETX                    ;
        MOV     AL,0BBH                  ;
        MOV     CL,AL                    ;
        CALL    KPREG                    ;
        AND     AL,ES:[BX]               ;
        CALL    KPREG                    ;
        CMP     AL,DL                    ;
        JNE     $+5                      ;
        JMP     BSETX                    ;
        MOV     AL,0EEH                  ;
        MOV     CL,AL                    ;
        CALL    KPREG                    ;
        AND     AL,ES:[BX]               ;
        CALL    KPREG                    ;
        CMP     AL,DL                    ;
        JNE     $+5                      ;
        JMP     BSETX                    ;

TYPRB:  MOV     CL,0                     ;
        MOV     BP,RED                   ;
        MOV     ES,BP                    ;
```

```
        MOV     AL,DL               ;} データ=FFHならRPRFへ
        CMP     AL,0FFH             ;
        JE      RPRF                ;

        CALL    CHKUD               ;} データの上位／下位が等しければRPRXへ
        JE      RPRX                ;

RCKE:   CMP     AL,DH               ;
        JNE     RNORM2              ;
        INC     CH                  ;
        MOV     AL,CH               ;} 上位／下位の違うデータが4桁以上連続
        CMP     AL,3                ;  した場合はSAMSTへ
        JNE     RNORM               ;
        JMP     SAMST               ;

RNORM2: MOV     CH,0                ;
RNORM:  MOV     [BX],DL             ;
        INC     BX                  ;
        CMP     BX,DTMAX            ;} 通常のデータ処理（アドレスがDTMAXを
        JB      $+5                 ;  越えたらMOVERへ）
        JMP     MOVER               ;
        MOV     DH,DL               ;
        JMP     RPRES               ;

TORPR:  MOV     DL,AL               ;
TORPR1: INC     BX                  ;
TORPR2: XOR     AL,AL               ;} 圧縮ルールからデータが連続した場合の処理
        MOV     CS:DCONT,AL         ;
        MOV     CH,AL               ;
        MOV     CL,AL               ;
        MOV     DH,AL               ;
        JMP     RPRE4               ;

RPRX:   MOV     [BX],DL             ;
        INC     BX                  ;
        MOV     BYTE PTR [BX],1     ;
        MOV     DH,DL               ;
        MOV     CL,DL               ;} 上位／下位が同じデータの2回目をチェック
        CALL    TDAT1               ;
        JB      $+5                 ;
        CALL    ROTER               ;
        MOV     DL,AL               ;
        CMP     AL,CL               ;
        JE      RXSSR1              ;
        CMP     AL,DH               ;
        JE      RXSS01              ;
        JMP     TORPR1              ;

RPR00:  MOV     BP,RED   ;data=00   ;
        MOV     ES,BP               ;
        MOV     [BX],AL             ;
        INC     BX                  ;
        MOV     BYTE PTR [BX],1     ;
        CALL    TDAT1               ;} データ=0の圧縮処理
        OR      AL,AL               ;
        JNE     TORPR               ;
RPR01:  CALL    DCON1               ;
        MOV     BYTE PTR [BX],2     ;
RXX0L:  CALL    TDATA               ;
        OR      AL,AL               ;
        JNE     TORPR               ;
        CALL    INC17               ;
        JMP     RXX0L               ;
```

```
RPRF:   MOV     [BX],AL               ;
        INC     BX                    ;
        MOV     BYTE PTR [BX],1       ;
        CALL    TDAT1                 ;
        CMP     AL,0FFH               ;
        JNE     TORPR                 ;
RPRF1:  CALL    DCON1                 ;  データ＝FFHの圧縮処理
        MOV     BYTE PTR [BX],2       ;
RXXFL:  CALL    TDATA                 ;
        CMP     AL,0FFH               ;
        JNE     TORPR                 ;
        CALL    INC1F                 ;
        JMP     RXXFL                 ;

RXSS0:  DEC     BX                    ;
        MOV     [BX],DH               ;
        INC     BX                    ;
        MOV     AL,DH                 ;
        INC     AL                    ;
        JE      RPRF1                 ;
        DEC     AL                    ;
        JE      RPR01                 ;  上位／下位が同じデータの圧縮処理
RXSS01: CALL    DCON1                 ;  （データはローテートしない）
        MOV     BYTE PTR [BX],82H     ;
RXS0L:  CALL    TDATA                 ;
        CMP     AL,DH                 ;
        JE      $+5                   ;
        JMP     TORPR                 ;
        CALL    INC8A                 ;
        JMP     RXS0L                 ;
RXSSR:  DEC     BX                    ;
        MOV     [BX],DH               ;
        INC     BX                    ;
RXSSR1: CALL    DCON1                 ;
        MOV     BYTE PTR [BX],2       ;
        MOV     CL,DL                 ;
RXSRL:  CALL    TDATA                 ;  上位／下位が同じデータの圧縮処理
        JB      $+5                   ;  （データはローテートする）
        CALL    ROTER                 ;
        CMP     AL,CL                 ;
        JE      $+5                   ;
        JMP     TORPR                 ;
        CALL    INC17                 ;
        JMP     RXSRL                 ;

BNXX0:  DEC     BX                    ;
        MOV     AH,CS:KEQB0           ;
        MOV     [BX],AH               ;
        INC     BX                    ;
        MOV     BYTE PTR [BX],0B2H    ;
        CALL    DCON1                 ;
BXLP0:  CALL    TDATA                 ;
        MOV     DL,AL                 ;
        MOV     AL,CL                 ;  データが「B面 AND CL」の場合の圧縮処理
        CALL    KPREG                 ;  （CLはローテートする）
        MOV     BP,BLUE               ;
        MOV     ES,BP                 ;
        AND     AL,ES:[BX]            ;
        MOV     BP,RED                ;
        MOV     ES,BP                 ;
        CALL    KPREG                 ;
        CMP     AL,DL                 ;
        JE      $+5                   ;
```

```
        JMP     TORPR1                  ;
        CALL    INCBB                   ;
        JMP     BXLP0                   ;

BNXXR:  DEC     BX                      ;
        MOV     AH,CS:KEQB1             ;
        MOV     [BX],AH                 ;
        INC     BX                      ;
        MOV     BYTE PTR [BX],0C2H      ;
        CALL    DCON1                   ;
BXLPR:  CALL    TDATA                   ;
        JB      $+5                     ;
        CALL    ROTER                   ;
        MOV     DL,AL                   ;
        MOV     AL,CL                   ;
        CALL    KPREG                   ;
        MOV     BP,BLUE                 ; データが「B面 AND CL」の場合の圧縮処理
        MOV     ES,BP                   ; (CLはローテートする)
        AND     AL,ES:[BX]              ;
        MOV     BP,RED                  ;
        MOV     ES,BP                   ;
        CALL    KPREG                   ;
        CMP     AL,DL                   ;
        JE      $+5                     ;
        JMP     TORPR1                  ;
        CALL    INCCF                   ;
        JMP     BXLPR                   ;

BSAME:  DEC     BX                      ;
        MOV     BYTE PTR [BX],0         ;
        INC     BX                      ;
        MOV     BYTE PTR [BX],82H       ;
        CALL    DCON1                   ;
BSMLP:  CALL    TDATA                   ;
        MOV     DL,AL                   ;
        CALL    KPREG                   ;
        MOV     BP,BLUE                 ; データがB面と同じ場合の圧縮処理
        MOV     ES,BP                   ;
        CMP     AL,ES:[BX]              ;
        MOV     BP,RED                  ;
        MOV     ES,BP                   ;
        CALL    KPREG                   ;
        JE      $+5                     ;
        JMP     TORPR1                  ;
        CALL    INC8F                   ;
        JMP     BSMLP                   ;

CFENT:  PUSH    BX                      ;
        MOV     BX,OFFSET RNEW          ;
        MOV     CS:KBFFP0,BX            ; B面と同じで上位/下位が違う場合の
        POP     BX                      ; 2回目のチェック
        MOV     AL,DL                   ;
        JMP     CFENT0                  ;

BSET0:  MOV     CL,0FFH                 ;
        MOV     BP,RED                  ; B面と同じで上位/下位が違う場合は
        MOV     ES,BP                   ; RCKEへ
        CALL    ABCHK                   ;
        JE      $+5                     ;
        JMP     RCKE                    ;
        PUSH    BX                      ;
        MOV     BX,OFFSET DATFF         ;
        MOV     CS:KBFFP0,BX            ;
```

```
        POP     BX                         ;
        MOV     DH,DL                      ;
        MOV     BYTE PTR [BX],0            ;
        INC     BX                         ;
        MOV     BYTE PTR [BX],81H          ;
        CALL    TDAT1                      ;  2回目のデータがB面と同じかどうかを
        MOV     DL,AL                      ;  チェックする
CFENT0: CALL    KPREG                      ;
        MOV     BP,BLUE                    ;
        MOV     ES,BP                      ;
        CMP     AL,ES:[BX]                 ;
        MOV     BP,RED                     ;
        MOV     ES,BP                      ;
        CALL    KPREG                      ;
        JE      BSAME                      ;
        JMP     CS:KBFFP0                  ;

DATFF:  CMP     AL,DH                      ;
        JNE     $+5                        ;
        JMP     RXSS0                      ;
        MOV     CL,DH                      ;
        CALL    ROTER                      ;  2回目のデータが前回（ローテートを含む）と
        CMP     AL,CL                      ;  同じかどうかをチェック
        JNE     $+5                        ;
        JMP     RXSSR                      ;
        INC     BX                         ;
        JMP     TORPR2                     ;

CXENT:  MOV     AL,90H                     ;
        MOV     CS:KTRP01,AL               ;
        PUSH    BX                         ;
        MOV     BX,OFFSET RNEW             ;  「B面 AND CL」と同じで上位／下位が
        MOV     CS:KBXXP0,BX               ;  違う場合の2回目をチェック
        POP     BX                         ;
        JMP     CXENT0                     ;
BSETX:  MOV     AL,CL                      ;
        MOV     CS:KEQB0,AL                ;
        MOV     CS:KEQB1,AL                ;
        MOV     BP,RED                     ;  「B面 AND CL」と同じで上位／下位が
        MOV     ES,BP                      ;  違う場合はRCKEへ
        MOV     AL,DL                      ;
        CALL    ABCHK                      ;
        JE      $+5                        ;
        JMP     RCKE                       ;
        PUSH    BX                         ;
        MOV     BX,OFFSET DATXX            ;
        MOV     CS:KBXXP0,BX               ;
        POP     BX                         ;
        MOV     AL,43H                     ;
        MOV     CS:KTRP01,AL               ;
        MOV     DH,DL                      ;
        MOV     [BX],CL                    ;
        INC     BX                         ;
        MOV     BYTE PTR [BX],0C1H         ;
        CALL    TDAT1                      ;
        JB      $+5                        ;
        CALL    ROTER                      ;
        MOV     DL,AL                      ;
CXENT0: MOV     AL,CL                      ;
        CALL    KPREG                      ;
        MOV     BP,BLUE                    ;
        MOV     ES,BP                      ;  2回目のデータが「B面 AND CL」（ローテート
        AND     AL,ES:[BX]                 ;  含む）と同じかどうかをチェック
```

```
         MOV     BP,RED                          ;
         MOV     ES,BP                           ;
         CALL    KPREG                           ;
         CMP     AL,DL                           ;
         JNE     $+5                             ;
         JMP     BNXXR                           ;
         CALL    ROTER                           ;
         MOV     AL,CL                           ;
         CALL    KPREG                           ;
         MOV     BP,BLUE                         ;
         MOV     ES,BP                           ;
         AND     AL,ES:[BX]                      ;
         MOV     BP,RED                          ;
         MOV     ES,BP                           ;
         CALL    KPREG                           ;
         CMP     AL,DL                           ;
         JNE     $+5                             ;
         JMP     BNXXO                           ;
         JMP     CS:KBXXPO                       ;

DATXX:   MOV     AL,DL                           ;
         CMP     AL,DH                           ;
         JNE     $+5                             ;
         JMP     RXSSO                           ;
         MOV     CL,DH                           ; 2回目のデータが前回（ローテート含む）と
         CALL    ROTER                           ; 同じかどうかをチェック
         CMP     AL,CL                           ;
         JNE     $+5                             ;
         JMP     RXSSR                           ;
KTRPO1   DB      43H                             ;
         JMP     TORPR2                          ;

GPRES:   MOV     AL,CL                           ;
         MOV     CS:KBROCK,AL                    ;
         MOV     CS:KRROCK,AL                    ;
         MOV     CS:KXXOCK,AL                    ;
         MOV     CS:KEQG0,AL                     ; プログラム上のワークエリアを初期化
         MOV     CS:KEQG1,AL                     ;
         MOV     CS:KEQG2,AL                     ;
         MOV     CS:KEQG3,AL                     ;
         MOV     CS:KEQG4,AL                     ;
         CALL    TDATA                           ; データ（G面）をリード同一縦列内ならCL
         JB      $+5                             ; レジスタをローテート
         CALL    ROTER                           ;
         MOV     DL,AL                           ;
GPRE4:   MOV     AL,CL                           ; CL=FFHならGRCFFへ
         INC     AL                              ;
         JE      GPCFF                           ;
         DEC     AL                              ; CL=00HならGNEWへ
         JE      GNEW                            ;
         TEST    CH,80H                          ;
         JE      $+5                             ; CHのビット7=1ならBRSCKへ
         JMP     BRSCK                           ;
         TEST    CH,40H                          ;
         JE      $+5                             ; CHのビット6=1ならBSAMへ
         JMP     BSAM                            ;
         JMP     RSAM                            ; CHのビット5=1ならRSAMへ

GPCFF:   MOV     AL,DL                           ;
         TEST    CH,80H                          ; CHのビット7=1ならCFFBRへ
         JE      $+5                             ;
         JMP     CFFBR                           ;
         TEST    CH,40H                          ;
         JE      $+5                             ; CHのビット6=1ならCFFBへ
```

```
        JMP     CFFB                  ;⌉
        JMP     CFFR                  ;  CHのビット5=1ならCFFRへ
                                      ;⌉
GNEW:   MOV     CL,0FFH               ;
        MOV     AL,DL                 ;
        CALL    KPREG                 ;
        MOV     BP,BLUE               ;
        MOV     ES,BP                 ;
        CMP     AL,ES:[BX]            ;
        JNE     $+5                   ;
        JMP     CKBRF                 ;
        MOV     BP,RED                ;
        MOV     ES,BP                 ;
        CMP     AL,ES:[BX]            ;
        CALL    KPREG                 ;
        JNE     $+5                   ;
        JMP     RSETF                 ;
        MOV     BP,BLUE               ;
        MOV     ES,BP                 ;
        MOV     AL,55H                ;
        MOV     CL,AL                 ;
        CALL    KPREG                 ;
        AND     AL,ES:[BX]            ;
        CALL    KPREG                 ;
        CMP     AL,DL                 ;
        JNE     $+5                   ;
        JMP     CKBRX                 ;
        MOV     AL,0AAH               ;
        MOV     CL,AL                 ;
        CALL    KPREG                 ;
        AND     AL,ES:[BX]            ;
        CALL    KPREG                 ;
        CMP     AL,DL                 ;
        JNE     $+5                   ;
        JMP     CKBRX                 ;
        MOV     BP,GREEN              ;
        MOV     ES,BP                 ;
        MOV     AL,DL                 ;
        OR      AL,AL                 ;
        JNE     $+5                   ;
        JMP     GDA0F                 ;
        MOV     BP,RED                ;
        MOV     ES,BP                 ;
        MOV     AL,55H                ;
        MOV     CL,AL                 ;
        CALL    KPREG                 ;
        AND     AL,ES:[BX]            ;
        CALL    KPREG                 ;
        CMP     AL,DL                 ;
        JNE     $+5                   ;
        JMP     RSETX                 ;
        MOV     AL,0AAH               ;
        MOV     CL,AL                 ;
        CALL    KPREG                 ;
        AND     AL,ES:[BX]            ;
        CALL    KPREG                 ;
        CMP     AL,DL                 ;
        JNE     $+5                   ;
        JMP     RSETX                 ;
        MOV     BP,BLUE               ;
        MOV     ES,BP                 ;
        MOV     AL,33H                ;
        MOV     CL,AL                 ;
```

```
        CALL    KPREG           ;
        AND     AL,ES:[BX]      ;
        CALL    KPREG           ;
        CMP     AL,DL           ;
        JNE     $+5             ;
        JMP     CKBRX           ;
        MOV     AL,66H          ;
        MOV     CL,AL           ;
        CALL    KPREG           ;
        AND     AL,ES:[BX]      ;
        CALL    KPREG           ;
        CMP     AL,DL           ;
        JNE     $+5             ;
        JMP     CKBRX           ;
        MOV     AL,99H          ; データ=0またはB/R面と共通(ANDも含む)
        MOV     CL,AL           ;
        CALL    KPREG           ;
        AND     AL,ES:[BX]      ;
        CALL    KPREG           ;
        CMP     AL,DL           ;
        JNE     $+5             ;
        JMP     CKBRX           ;
        MOV     AL,0CCH         ;
        MOV     CL,AL           ;
        CALL    KPREG           ;
        AND     AL,ES:[BX]      ;
        CALL    KPREG           ;
        CMP     AL,DL           ;
        JNE     $+5             ;
        JMP     CKBRX           ;
        MOV     BP,RED          ;
        MOV     ES,BP           ;
        MOV     AL,33H          ;
        MOV     CL,AL           ;
        CALL    KPREG           ;
        AND     AL,ES:[BX]      ;
        CALL    KPREG           ;
        CMP     AL,DL           ;
        JNE     $+5             ;
        JMP     RSETX           ;
        MOV     AL,66H          ;
        MOV     CL,AL           ;
        CALL    KPREG           ;
        AND     AL,ES:[BX]      ;
        CALL    KPREG           ;
        CMP     AL,DL           ;
        JNE     $+5             ;
        JMP     RSETX           ;
        MOV     AL,99H          ;
        MOV     CL,AL           ;
        CALL    KPREG           ;
        AND     AL,ES:[BX]      ;
        CALL    KPREG           ;
        CMP     AL,DL           ;
        JNE     $+5             ;
        JMP     RSETX           ;
        MOV     AL,0CCH         ;
        MOV     CL,AL           ;
        CALL    KPREG           ;
        AND     AL,ES:[BX]      ;
        CALL    KPREG           ;
        CMP     AL,DL           ;
        JNE     $+5             ;
```

```
        JMP     RSETX                           ;
        JMP     CS:KTYPEG                       ;

TYPGA:  MOV     AL,22H                          ;
        MOV     BP,BLUE                         ;
        MOV     ES,BP                           ;
        MOV     CL,AL                           ;
        CALL    KPREG                           ;
        AND     AL,ES:[BX]                      ;
        CALL    KPREG                           ;
        CMP     AL,DL                           ;
        JNE     $+5                             ;
        JMP     CKBRX                           ;
        MOV     AL,77H                          ;
        MOV     CL,AL                           ;
        CALL    KPREG                           ;
        AND     AL,ES:[BX]                      ;
        CALL    KPREG                           ;
        CMP     AL,DL                           ;
        JNE     $+5                             ;
        JMP     CKBRX                           ;
        MOV     AL,88H                          ;
        MOV     CL,AL                           ;
        CALL    KPREG                           ;
        AND     AL,ES:[BX]                      ;
        CALL    KPREG                           ;
        CMP     AL,DL                           ;
        JNE     $+5                             ;
        JMP     CKBRX                           ;
        MOV     AL,0DDH                         ;
        MOV     CL,AL                           ;
        CALL    KPREG                           ;
        AND     AL,ES:[BX]                      ;
        CALL    KPREG                           ;
        CMP     AL,DL                           ;
        JNE     $+5                             ;
        JMP     CKBRX                           ;
                                                ;
        MOV     BP,RED                          ;
        MOV     ES,BP                           ;
        MOV     AL,22H                          ;
        MOV     CL,AL                           ;
        CALL    KPREG                           ;
        AND     AL,ES:[BX]                      ;
        CALL    KPREG                           ;
        CMP     AL,DL                           ;
        JNE     $+5                             ;
        JMP     RSETX                           ;
        MOV     AL,77H                          ;
        MOV     CL,AL                           ;
        CALL    KPREG                           ;
        AND     AL,ES:[BX]                      ;
        CALL    KPREG                           ;
        CMP     AL,DL                           ;
        JNE     $+5                             ;
        JMP     RSETX                           ;
        MOV     AL,88H                          ;
        MOV     CL,AL                           ;
        CALL    KPREG                           ;
        AND     AL,ES:[BX]                      ;
        CALL    KPREG                           ;
        CMP     AL,DL                           ;
        JNE     $+5                             ;
```

```
        JMP     RSETX           ;
        MOV     AL,0DDH         ;
        MOV     CL,AL           ;
        CALL    KPREG           ;
        AND     AL,ES:[BX]      ;  B／R面と共通（ANDも含む）ならそれぞ
        CALL    KPREG           ;  れのルーチンへ
        CMP     AL,DL           ;
        JNE     $+5             ;
        JMP     RSETX           ;
                                ;
        MOV     BP,BLUE         ;
        MOV     ES,BP           ;
        MOV     AL,11H          ;
        MOV     CL,AL           ;
        CALL    KPREG           ;
        AND     AL,ES:[BX]      ;
        CALL    KPREG           ;
        CMP     AL,DL           ;
        JNE     $+5             ;
        JMP     CKBRX           ;
        MOV     AL,44H          ;
        MOV     CL,AL           ;
        CALL    KPREG           ;
        AND     AL,ES:[BX]      ;
        CALL    KPREG           ;
        CMP     AL,DL           ;
        JNE     $+5             ;
        JMP     CKBRX           ;
        MOV     AL,0BBH         ;
        MOV     CL,AL           ;
        CALL    KPREG           ;
        AND     AL,ES:[BX]      ;
        CALL    KPREG           ;
        CMP     AL,DL           ;
        JNE     $+5             ;
        JMP     CKBRX           ;
        MOV     AL,0EEH         ;
        MOV     CL,AL           ;
        CALL    KPREG           ;
        AND     AL,ES:[BX]      ;
        CALL    KPREG           ;
        CMP     AL,DL           ;
        JNE     $+5             ;
        JMP     CKBRX           ;
                                ;
        MOV     BP,RED          ;
        MOV     ES,BP           ;
        MOV     AL,11H          ;
        MOV     CL,AL           ;
        CALL    KPREG           ;
        AND     AL,ES:[BX]      ;
        CALL    KPREG           ;
        CMP     AL,DL           ;
        JNE     $+5             ;
        JMP     RSETX           ;
        MOV     AL,44H          ;
        MOV     CL,AL           ;
        CALL    KPREG           ;
        AND     AL,ES:[BX]      ;
        CALL    KPREG           ;
        CMP     AL,DL           ;
        JNE     $+5             ;
        JMP     RSETX           ;
```

```
        MOV     AL,0BBH             ;
        MOV     CL,AL               ;
        CALL    KPREG               ;
        AND     AL,ES:[BX]          ;
        CALL    KPREG               ;
        CMP     AL,DL               ;
        JNE     $+5                 ;
        JMP     RSETX               ;
        MOV     AL,0EEH             ;
        MOV     CL,AL               ;
        CALL    KPREG               ;
        AND     AL,ES:[BX]          ;
        CALL    KPREG               ;
        CMP     AL,DL               ;
        JNE     $+5                 ;
        JMP     RSETX               ;

TYPGB:  MOV     AL,CH               ;
        MOV     CL,0                ;
        MOV     BP,GREEN            ;
        MOV     ES,BP               ;
        AND     AL,0FH              ; データ=FFHまたは上位／下位が等しければ
        MOV     CH,AL               ; それぞれのルーチンへ
        MOV     AL,DL               ;
        CMP     AL,0FFH             ;
        JE      GDA0F               ;
        CALL    CHKUD               ;
        JE      GDAXX               ;
GCKE:   CMP     AL,DH               ;
        JNE     GNOR0               ;
        INC     CH                  ;
        MOV     AL,CH               ; 同じデータが4ケ以上連続した場合は
        AND     AL,0FH              ; SAMTへ
        CMP     AL,3                ;
        JNE     GNORM               ;
        JMP     SAMST               ;
GNOR0:  AND     CH,0FEH             ;
        AND     CH,0FDH             ;
GNORM:  MOV     [BX],DL             ;
        INC     BX                  ; 通常のデータ処理（アドレスがDTMAXを
        CMP     BX,DTMAX            ; 越えたらMOVERへ）
        JB      $+5                 ;
        JMP     MOVER               ;
        MOV     DH,DL               ;
        JMP     GPRES               ;

GDA0F:  MOV     [BX],AL             ;
        INC     BX                  ;
        MOV     BYTE PTR [BX],1     ;
        CALL    DCON1               ; データ=00HまたはFFHの場合の圧縮処理
GSAML:  CALL    TDATA               ;
        CMP     AL,DL               ;
        JNE     TOGPR               ;
GUP0F:  CALL    INC17               ;
        JMP     GSAML               ;
TOGPR:  MOV     DL,AL               ;
TOGPR1: INC     BX                  ;
        XOR     AL,AL               ;
        MOV     CS:DCONT,AL         ;
        MOV     CH,AL               ; 圧縮ルールからデータが逸脱した場合の処理
        MOV     CL,AL               ;
        MOV     DH,AL               ;
        JMP     GPRE4               ;
```

```
ROTER:  PUSH    AX                  ;
        MOV     AL,CL               ;
        CMP     AL,55H              ;
        JE      ROT11               ;
        CMP     AL,0AAH             ;  CL レジスタの値をローテートする
        JE      ROT11               ;
        ROR     CL,1                ;
ROT11:  ROR     CL,1                ;
        POP     AX                  ;
        RET                         ;

KPOINT  DW      OFFSET BPRES        ;
KRETAD  DW      OFFSET TOBPR        ;
KTATE4  DW      OFFSET ATOP         ;
KTYPER  DW      OFFSET TYPRA        ;  ワーク・エリア
KTYPEG  DW      OFFSET TYPGA        ;
KPLANE  DW      BLUE                ;
KEQB0   DB      0                   ;
KEQB1   DB      0                   ;
KBFFPO  DW      OFFSET  DATFF       ;
KBXXPO  DW      OFFSET  DATXX       ;

GDAXX:  MOV     [BX],DL             ;
        INC     BX                  ;
        MOV     BYTE PTR [BX],1     ;
        MOV     DH,DL               ;
        MOV     CL,DL               ;  2回目のデータチェック
        CALL    TDAT1               ;  1回目＝上位／下位が同じ
        JB      $+5                 ;
        CALL    ROTER               ;
        MOV     DL,AL               ;
        CMP     AL,CL               ;
        JNE     GSACK               ;

SROTX2: MOV     BYTE PTR [BX],2     ;
        CALL    DCON1               ;
ROTXX:  CALL    TDATA               ;
        JB      $+5                 ;
        CALL    ROTER               ;  上位／下位が同じデータの圧縮処理
        MOV     DL,AL               ;  （データはローテートする）
        CMP     AL,CL               ;
        JNE     TOGPR1              ;
        CALL    INC13               ;
        JMP     ROTXX               ;

GSACK:  CMP     AL,DH               ;
        JNE     TOGPR1              ;
        JMP     SSAMX2              ;
SAMFX:  CALL    DCON1               ;
        MOV     AL,DL               ;
        INC     AL                  ;
        JE      GUPOF               ;  上位／下位が同じデータの圧縮処理
        DEC     AL                  ;  （データはローテートしない）
        JE      GUPOF               ;
SSAMX2: MOV     BYTE PTR [BX],42H   ;
        CALL    DCON1               ;
SAMXX:  CALL    TDATA               ;
        CMP     AL,DH               ;
        JE      $+5                 ;
        JMP     TOGPR               ;
        CALL    INC45               ;
        JMP     SAMXX               ;
```

```
SBXX0:  MOV     BYTE PTR [BX],62H    ;
        CALL    DCON1                ;
BXXL0:  CALL    TDATA                ;
        MOV     DL,AL                ;
        MOV     AL,CL                ;
        CALL    KPREG                ;
        MOV     BP,BLUE              ; データが「B面 AND CL」の場合の圧縮処理
        MOV     ES,BP                ; (CLはローテートしない)
        AND     AL,ES:[BX]           ;
        MOV     BP,GREEN             ;
        MOV     ES,BP                ;
        CALL    KPREG                ;
        CMP     AL,DL                ;
        JE      $+5                  ;
        JMP     TOGPR1               ;
BSSTX:  CALL    INC66                ;
        JMP     BXXL0                ;

SRXX0:  MOV     BYTE PTR [BX],72H    ;
        CALL    DCON1                ;
RXXL0:  CALL    TDATA                ;
        MOV     DL,AL                ;
CRAND0: MOV     AL,CL                ;
        CALL    KPREG                ;
        MOV     BP,RED               ;
        MOV     ES,BP                ; データが「R面 AND CL」の場合の圧縮処理
        AND     AL,ES:[BX]           ; (CLはローテートしない)
        MOV     BP,GREEN             ;
        MOV     ES,BP                ;
        CALL    KPREG                ;
        CMP     AL,DL                ;
        JE      $+5                  ;
        JMP     TOGPR1               ;
        CALL    INC77                ;
        JMP     RXXL0                ;
SBXXR:  MOV     BYTE PTR [BX],82H    ;
        CALL    DCON1                ;
BXXLR:  CALL    TDATA                ;
        JB      $+5                  ;
        CALL    ROTER                ;
        MOV     DL,AL                ;
        MOV     AL,CL                ; データが「B面 AND CL」の場合の圧縮処理
        CALL    KPREG                ; (CLはローテートする)
        MOV     BP,BLUE              ;
        MOV     ES,BP                ;
        AND     AL,ES:[BX]           ;
        MOV     BP,GREEN             ;
        MOV     ES,BP                ;
        CALL    KPREG                ;
        CMP     AL,DL                ;
        JE      $+5                  ;
        JMP     TOGPR1               ;
BRSTX:  CALL    INC8B                ;
        JMP     BXXLR                ;

SRXXR:  MOV     BYTE PTR [BX],0C2H   ;
        CALL    DCON1                ;
RXXLR:  CALL    TDATA                ;
        JB      $+5                  ;
        CALL    ROTER                ;
        MOV     DL,AL                ; データが「R面 AND CL」の場合の圧縮処理
CRANDR: MOV     AL,CL                ; (CLはローテートする)
        CALL    KPREG                ;
```

```
        MOV     BP,RED                  ;
        MOV     ES,BP                   ;
        AND     AL,ES:[BX]              ;
        MOV     BP,GREEN                ;
        MOV     ES,BP                   ;
        CALL    KPREG                   ;
        CMP     AL,DL                   ;
        JE      $+5                     ;
        JMP     TOGPR1                  ;
        CALL    INCCF                   ;
        JMP     SHORT RXXLR             ;

WWXX0:  MOV     BYTE PTR [BX],62H       ;
        CALL    DCON1                   ;
WXXL0:  CALL    TDATA                   ;
        MOV     DL,AL                   ;
        MOV     AL,CL                   ;
        CALL    KPREG                   ;
        MOV     BP,BLUE                 ;
        MOV     ES,BP                   ;
        AND     AL,ES:[BX]              ;
        MOV     BP,GREEN                ;
        MOV     ES,BP                   ;
        CALL    KPREG                   ;
        CMP     AL,DL                   ;
        JNE     TRCON                   ;
        MOV     AL,CL                   ;
        CALL    KPREG                   ;
        MOV     BP,RED                  ;  データが「B面 AND CL」「R面 AND CL」と
        MOV     ES,BP                   ;  同じ場合（CLはローテートしない）
        AND     AL,ES:[BX]              ;  長く連続するほうを求める
        MOV     BP,GREEN                ;
        MOV     ES,BP                   ;
        CALL    KPREG                   ;
        CMP     AL,DL                   ;
        JE      $+5                     ;
        JMP     BSSTX                   ;
        CALL    INC66                   ;
        JMP     SHORT WXXL0             ;
TRCON:  PUSH    BX                      ;
        MOV     AL,CS:DCONT             ;
        DEC     AL                      ;
        JE      TRDC0                   ;
        DEC     BX                      ;
        DEC     BX                      ;
TRDC0:  OR      BYTE PTR [BX],10H       ;
        POP     BX                      ;
        JMP     CRAND0                  ;

WWXXR:  MOV     BYTE PTR [BX],82H       ;
        CALL    DCON1                   ;
WXXLR:  CALL    TDATA                   ;
        JB      $+5                     ;
        CALL    ROTER                   ;
        MOV     DL,AL                   ;
        MOV     AL,CL                   ;
        CALL    KPREG                   ;
        MOV     BP,BLUE                 ;
        MOV     ES,BP                   ;
        AND     AL,ES:[BX]              ;
        MOV     BP,GREEN                ;
        MOV     ES,BP                   ;
        CALL    KPREG                   ;
```

```
        CMP     AL,DL              ;
        JNE     TRCON2             ;
        MOV     AL,CL              ;
        CALL    KPREG              ;  データが「B面 AND CL」「R面 AND CL」と
        MOV     BP,RED             ;  同じ場合（CLはローテートする）
        MOV     ES,BP              ;  長く連続するほうを求める
        AND     AL,ES:[BX]         ;
        MOV     BP,GREEN           ;
        MOV     ES,BP              ;
        CALL    KPREG              ;
        CMP     AL,DL              ;
        JE      $+5                ;
        JMP     BRSTX              ;
        CALL    INC8B              ;
        JMP     SHORT WXXLR        ;
TRCON2: PUSH    BX                 ;
        MOV     AL,CS:DCONT        ;
        DEC     AL                 ;
        JE      TRDC00             ;
        DEC     BX                 ;
        DEC     BX                 ;
TRDC00: OR      BYTE PTR [BX],40H  ;
        POP     BX                 ;
        JMP     CRANDR             ;

BSAM:   MOV     AL,CH              ;
        AND     AL,0FH             ;
        MOV     CH,AL              ;
        MOV     AL,CL              ;
        CALL    KPREG              ;
        MOV     BP,BLUE            ;
        MOV     ES,BP              ;
        MOV     BP,BLUE            ;
        MOV     ES,BP              ;
        AND     AL,ES:[BX]         ;  2回目のデータ・チェック（CLはローテートする）
        MOV     BP,GREEN           ;  1回目＝「B面 AND CL」と同じ、上位／下位
        MOV     ES,BP              ;  は違う
        CALL    KPREG              ;
        CMP     AL,DL              ;
        JNE     JBROCK             ;
        DEC     BX                 ;
        MOV     AH,CS:KEQG0        ;
        MOV     [BX],AH            ;
        INC     BX                 ;
        JMP     SBXXR              ;

JBROCK: MOV     CL,CS:KBROCK       ;
        MOV     AL,CL              ;
        CALL    KPREG              ;
        MOV     BP,BLUE            ;
        MOV     ES,BP              ;
        AND     AL,ES:[BX]         ;
        CALL    KPREG              ;  2回目のデータチェック(CLはローテートしない)
        MOV     BP,GREEN           ;  1回目＝「B面 AND CL」と同じ、上位／下位
        MOV     ES,BP              ;  は違う
        CMP     AL,DL              ;
        JE      $+5                ;
        JMP     GNEW               ;
        DEC     BX                 ;
        MOV     [BX],CL            ;
        INC     BX                 ;
        JMP     SBXX0              ;
```

```
RSAM:    MOV     AL,CH          ;
         AND     AL,0FH         ;
         MOV     CH,AL          ;
         MOV     AL,CL          ;
         CALL    KPREG          ;
         MOV     BP,RED         ;
         MOV     ES,BP          ;
         AND     AL,ES:[BX]     ; 2回目のデータチェック(CLはローテートする)
         MOV     BP,GREEN       ; 1回目=「R面 AND CL」と同じ、上位／下位
         MOV     ES,BP          ; は違う
         CALL    KPREG          ;
         CMP     AL,DL          ;
         JNE     JRROCK         ;
         DEC     BX             ;
         MOV     AH,CS:KEQG1    ;
         MOV     [BX],AH        ;
         INC     BX             ;
         JMP     SRXXR          ;

JRROCK:  MOV     CL,CS:KRROCK   ;
         MOV     AL,CL          ;
         CALL    KPREG          ;
         MOV     BP,RED         ;
         MOV     ES,BP          ;
         AND     AL,ES:[BX]     ;
         MOV     BP,GREEN       ; 2回目のデータチェック(CLはローテートしない)
         MOV     ES,BP          ; 1回目=「R面 AND CL」と同じ、上位／下位
         CALL    KPREG          ; は違う
         CMP     AL,DL          ;
         JE      $+5            ;
         JMP     GNEW           ;
         DEC     BX             ;
         MOV     [BX],CL        ;
         INC     BX             ;
         JMP     SRXXO

BRSCK:   MOV     AL,CH          ;
         AND     AL,0FH         ;
         MOV     CH,AL          ;
         MOV     AL,CL          ;
         MOV     BP,BLUE        ;
         MOV     ES,BP          ;
         CALL    KPREG          ;
         AND     AL,ES:[BX]     ;
         CALL    KPREG          ;
         CMP     AL,DL          ; 2回目のデータチェック(CLはローテートする)
         JNE     SCKRM          ; 1回目=「B／R面 AND CL」と同じ、
         MOV     AL,CL          ; 上位／下位は違う
         CALL    KPREG          ; 2回目=「B面 AND CL」でない場合
         MOV     BP,RED         ;
         MOV     ES,BP          ;
         AND     AL,ES:[BX]     ;
         MOV     BP,GREEN       ;
         MOV     ES,BP          ;
         CALL    KPREG          ;
         CMP     AL,DL          ;
         JE      TWXXR          ;
         DEC     BX             ;
         MOV     AH,CS:KEQG2    ;
         MOV     [BX],AH        ;
         INC     BX             ;
         JMP     SBXXR          ;
TWXXR:   DEC     BX
```

```
        MOV     AH,CS:KEQG3             ;
        MOV     [BX],AH                 ;
        INC     BX                      ;
        JMP     WWXXR                   ;

SCKRM:  MOV     AL,CL                   ;
        CALL    KPREG                   ;
        MOV     BP,RED                  ;
        MOV     ES,BP                   ;
        AND     AL,ES:[BX]              ;
        MOV     BP,GREEN                ;
        MOV     ES,BP                   ;
        CALL    KPREG                   ;  2回目のデータチェック(CLはローテートする)
        CMP     AL,DL                   ;  1回目＝「B／R面 AND CL」と同じ、上位
        JNE     JXXOCK                  ;  ／下位は違う
        DEC     BX                      ;  2回目＝「B面 AND CL」でない場合
        MOV     AH,CS:KEQG4             ;
        MOV     [BX],AH                 ;
        INC     BX                      ;
        JMP     SRXXR                   ;

JXXOCK: MOV     CL,CS:KXXOCK            ;
        MOV     AL,CL                   ;
        CALL    KPREG                   ;
        MOV     BP,BLUE                 ;
        MOV     ES,BP                   ;
        AND     AL,ES:[BX]              ;
        CALL    KPREG                   ;
        CMP     AL,DL                   ;
        JE      $+5                     ;
        JMP     JRROCK                  ;
        DEC     BX                      ;  2回目のデータチェック(CLはローテートしない)
        MOV     [BX],CL                 ;  1回目＝「B／R面 AND CL」と同じ、上位
        INC     BX                      ;  ／下位は違う
        MOV     AL,CL                   ;
        CALL    KPREG                   ;
        MOV     BP,RED                  ;
        MOV     ES,BP                   ;
        AND     AL,ES:[BX]              ;
        MOV     BP,GREEN                ;
        MOV     ES,BP                   ;
        CALL    KPREG                   ;
        CMP     AL,DL                   ;
        JNE     $+5                     ;
        JMP     WWXXO                   ;
        JMP     SBXXO                   ;

BSETXX: MOV     BP,GREEN                ;
        MOV     ES,BP                   ;
        MOV     AL,DL                   ;  データ＝00Hの場合はGDA0Fへ
        OR      AL,AL                   ;
        JNE     $+5                     ;
        JMP     GDA0F                   ;

        OR      CH,40H                  ;
        CALL    CHKUD                   ;  上位≠下位の場合はGCKEへ
        JE      $+5                     ;
        JMP     GCKE                    ;

        MOV     DH,DL                   ;
        MOV     [BX],CL                 ;
        INC     BX                      ;
        MOV     BYTE PTR [BX],81H       ;
```

```
        CALL    TDAT1           ;
        JB      $+5             ;
        CALL    ROTER           ;
        MOV     DL,AL           ;
        MOV     AL,CL           ;
        CALL    KPREG           ;
        MOV     BP,BLUE         ;
        MOV     ES,BP           ;
        AND     AL,ES:[BX]      ;
        MOV     BP,GREEN        ;
        MOV     ES,BP           ;
        CALL    KPREG           ;
        CMP     AL,DL           ;
        JNE     $+5             ;  2回目のデータチェック
        JMP     SBXXR           ;  1回目＝「B面 AND CL」と同じ、上位
        CALL    ROTER           ;  ／下位は同じ
        MOV     AL,CL           ;
        CALL    KPREG           ;
        MOV     BP,BLUE         ;
        MOV     ES,BP           ;
        AND     AL,ES:[BX]      ;
        MOV     BP,GREEN        ;
        MOV     ES,BP           ;
        CALL    KPREG           ;
        CMP     AL,DL           ;
        JNE     $+5             ;
        JMP     SBXX0           ;
CONCK:  DEC     BX              ;
        MOV     [BX],DH         ;
        INC     BX              ;
        MOV     BYTE PTR [BX],1 ;
        MOV     AL,DL           ;
        CMP     AL,DH           ;
        JNE     $+5             ;
        JMP     SAMFX           ;
        CALL    ROTER           ;
        MOV     AL,CL           ;
        CMP     AL,DL           ;
        JNE     $+5             ;
        JMP     SROTX2          ;
        JMP     TOGPR1          ;

RSETX:  OR      CH,20H          ;
        MOV     BP,GREEN        ;
        MOV     ES,BP           ;  上位≠下位の場合はGCKEへ
        CALL    ABCHK           ;
        JE      $+5             ;
        JMP     GCKE            ;

        MOV     DH,DL           ;
        MOV     [BX],CL         ;
        INC     BX              ;
        MOV     BYTE PTR [BX],0C1H ;
        CALL    TDAT1           ;
        JB      $+5             ;
        CALL    ROTER           ;
        MOV     DL,AL           ;
        MOV     AL,CL           ;
        CALL    KPREG           ;  2回目のデータチェック
        MOV     BP,RED          ;  1回目＝「R面 AND CL」と同じ、上位
        MOV     ES,BP           ;  ／下位は同じ
        AND     AL,ES:[BX]      ;
        MOV     BP,GREEN
```

```
        MOV     ES,BP               ;
        CALL    KPREG               ;
        CMP     AL,DL               ;
        JNE     $+5                 ;
        JMP     SRXXR               ;
        CALL    ROTER               ;
        MOV     AL,CL               ;
        CALL    KPREG               ;
        MOV     BP,RED              ;
        MOV     ES,BP               ;
        AND     AL,ES:[BX]          ;
        MOV     BP,GREEN            ;
        MOV     ES,BP               ;
        CALL    KPREG               ;
        CMP     AL,DL               ;
        JNE     $+5                 ;
        JMP     SRXXO               ;
        JMP     SHORT CONCK         ;

CKBRX:  MOV     AL,CL               ;
        CALL    KPREG               ;
        MOV     BP,RED              ;
        MOV     ES,BP               ;
        AND     AL,ES:[BX]          ;
        CALL    KPREG               ;
        CMP     AL,DL               ;
        JE      $+5                 ;
        JMP     BSETXX              ;
        OR      CH,80H              ;
        MOV     BP,GREEN            ;
        MOV     ES,BP               ;
        CALL    ABCHK               ;
        JE      $+5                 ;
        JMP     GCKE                ;
        MOV     DH,DL               ;
        MOV     [BX],CL             ;
        INC     BX                  ;
        MOV     BYTE PTR [BX],81H   ;
        CALL    TDAT1               ;
        JB      $+5                 ;
        CALL    ROTER               ;
        MOV     DL,AL               ;
        MOV     AL,CL               ;
        CALL    KPREG               ;
        MOV     BP,BLUE             ;
        MOV     ES,BP               ;
        AND     AL,ES:[BX]          ;
        MOV     BP,GREEN            ;
        MOV     ES,BP               ;
        CALL    KPREG               ;
        CMP     AL,DL               ;
        JNE     CKRBR               ;
        MOV     AL,CL               ;
        CALL    KPREG               ;
        MOV     BP,RED              ;
        MOV     ES,BP               ;
        AND     AL,ES:[BX]          ;
        MOV     BP,GREEN            ;
        MOV     ES,BP               ;
        CALL    KPREG               ;
        CMP     AL,DL               ;
        JE      $+5                 ;
        JMP     SBXXR               ;
```

```
         JMP     WWXXR                  ;
CKRBR:   MOV     AL,CL                  ;
         CALL    KPREG                  ;} 2回目のデータチェック
         MOV     BP,RED                 ;  1回目＝「B／R面 AND CL」と同じ、上位
         MOV     ES,BP                  ;  ／下位は同じ
         AND     AL,ES:[BX]             ;
         MOV     BP,GREEN               ;
         MOV     ES,BP                  ;
         CALL    KPREG                  ;
         CMP     AL,DL                  ;
         JNE     $+5                    ;
         JMP     SRXXR                  ;
         CALL    ROTER                  ;
         MOV     AL,CL                  ;
         CALL    KPREG                  ;
         MOV     BP,BLUE                ;
         MOV     ES,BP                  ;
         AND     AL,ES:[BX]             ;
         MOV     BP,GREEN               ;
         MOV     ES,BP                  ;
         CALL    KPREG                  ;
         CMP     AL,DL                  ;
         JNE     CKRB0                  ;
         MOV     AL,CL                  ;
         CALL    KPREG                  ;
         MOV     BP,RED                 ;
         MOV     ES,BP                  ;
         AND     AL,ES:[BX]             ;
         MOV     BP,GREEN               ;
         MOV     ES,BP                  ;
         CALL    KPREG                  ;
         CMP     AL,DL                  ;
         JE      $+5                    ;
         JMP     SBXX0                  ;
         JMP     WWXX0                  ;
CKRB0:   MOV     AL,CL                  ;
         CALL    KPREG                  ;
         MOV     BP,RED                 ;
         MOV     ES,BP                  ;
         AND     AL,ES:[BX]             ;
         MOV     BP,GREEN               ;
         MOV     ES,BP                  ;
         CALL    KPREG                  ;
         CMP     AL,DL                  ;
         JNE     $+5                    ;
         JMP     SRXX0                  ;
         JMP     CONCK                  ;

CFFR:    AND     CH,0DFH                ;
         CALL    KPREG                  ;
RCF00:   MOV     BP,RED                 ;
         MOV     ES,BP                  ;
         CMP     AL,ES:[BX]             ;
         MOV     BP,GREEN               ;
         MOV     ES,BP                  ;} 2回目のデータチェック
         CALL    KPREG                  ;  1回目＝R面と同じ、上位／下位は違う
         JE      $+5                    ;
         JMP     GNEW                   ;
         DEC     BX                     ;
         MOV     BYTE PTR [BX],0FFH     ;
         INC     BX                     ;
         MOV     BYTE PTR [BX],82H      ;
         JMP     SHORT RFST             ;
```

```
RSETF:  MOV     BP,GREEN                ;
        MOV     ES,BP                   ;  データ=00Hの場合はGDA0Fへ
        OR      AL,AL                   ;
        JNE     $+5                     ;
        JMP     GDA0F                   ;

        OR      CH,20H                  ;
        CALL    ABCK2                   ;  上位≠下位の場合はGCKEへ
        JE      $+5                     ;
        JMP     GCKE                    ;

        MOV     〔BX〕,CL                ;
        INC     BX                      ;
        MOV     BYTE PTR 〔BX〕,81H      ;
RFST:   CALL    DCON1                   ;
RFLP:   CALL    TDATA                   ;
RSSTF:  CALL    KPREG                   ;
        MOV     BP,RED                  ;
        MOV     ES,BP                   ;
        CMP     AL,ES:〔BX〕             ;  2回目以降のデータチェック
        MOV     BP,GREEN                ;  1回目=「R面 AND CL」と同じ、上位/下位
        MOV     ES,BP                   ;  は同じ
        CALL    KPREG                   ;
        JE      $+5                     ;
        JMP     TOGPR                   ;
        CALL    INC8F                   ;
        JMP     SHORT RFLP              ;

CFFBR:  AND     CH,07FH                 ;
        CALL    KPREG                   ;
        MOV     BP,BLUE                 ;
        MOV     ES,BP                   ;
        CMP     AL,ES:〔BX〕             ;
        JNE     RCF00                   ;
        MOV     BP,RED                  ;
        MOV     ES,BP                   ;
        CMP     AL,ES:〔BX〕             ;
        MOV     BP,GREEN                ;
        MOV     ES,BP                   ;  2回目のデータチェック
        CALL    KPREG                   ;  1回目=B/R面と同じ、上位/下位は違う
        JE      $+5                     ;
        JMP     STBSM                   ;
        DEC     BX                      ;
        MOV     BYTE PTR 〔BX〕,0FFH     ;
        INC     BX                      ;
        MOV     BYTE PTR 〔BX〕,82H      ;
        CALL    DCON1                   ;
        JMP     SHORT BRFL              ;

CKBRF:  MOV     BP,RED                  ;
        MOV     ES,BP                   ;
        CMP     AL,ES:〔BX〕             ;
        CALL    KPREG                   ;
        MOV     BP,GREEN                ;  データがB面と同じでR面と違う場合はBSTF
        MOV     ES,BP                   ;
        JE      $+5                     ;
        JMP     BSTF                    ;

        OR      CH,80H                  ;
        CALL    ABCHK                   ;  上位≠下位の場合はGCKEへ
        JE      $+5                     ;
        JMP     GCKE                    ;
```

```
        MOV     [BX],CL                ;
        INC     BX                     ;
        MOV     BYTE PTR [BX],81H      ;
        CALL    DCON1                  ;
BRFL:   CALL    TDATA                  ;
        CALL    KPREG                  ;
        MOV     BP,BLUE                ;
        MOV     ES,BP                  ;
        CMP     AL,ES:[BX]             ;
        CALL    KPREG                  ;
        JE      $+5                    ;  データがB／R面と同じ場合長く連続する
        JMP     RSSTF                  ;  ほうを求める
        MOV     BP,RED                 ;
        MOV     ES,BP                  ;
        CALL    KPREG                  ;
        CMP     AL,ES:[BX]             ;
        CALL    KPREG                  ;
        JNE     BSSTF                  ;
        MOV     BP,GREEN               ;
        MOV     ES,BP                  ;
        CALL    INC8F                  ;
        JMP     BRFL                   ;

BSSTF:  PUSH    BX                     ;
        MOV     DL,AL                  ;
        MOV     AL,CS:DCONT            ;
        DEC     AL                     ;
        JE      BSFD1                  ;
        DEC     BX                     ;  データが「B／R面と同じ」から「B面と同じ」
        DEC     BX                     ;  へ決定した場合の圧縮処理
BSFD1:  DEC     BX                     ;
        INC     BYTE PTR [BX]          ;
        POP     BX                     ;
        MOV     AL,DL                  ;
        JMP     BFENT                  ;

CFFB:   AND     CH,0BFH                ;
        CALL    KPREG                  ;
        MOV     BP,BLUE                ;
        MOV     ES,BP                  ;
        CMP     AL,ES:[BX]             ;
        MOV     BP,GREEN               ;
        MOV     ES,BP                  ;  2回目のデータチェック1回目＝B面と同じ
        CALL    KPREG                  ;  上位／下位は違う
        JE      $+5                    ;
        JMP     GNEW                   ;
STBSM:  DEC     BX                     ;
        MOV     BYTE PTR [BX],0        ;
        INC     BX                     ;
        MOV     BYTE PTR [BX],82H      ;
        CALL    DCON1                  ;
        JMP     SHORT BSFL             ;

BSTF:   OR      CH,40H                 ;
        MOV     BP,GREEN               ;
        MOV     ES,BP                  ;  データがB面と同じで上位≠下位の場合は
        CALL    ABCHK                  ;  GCKEへ
        JE      $+5                    ;
        JMP     GCKE                   ;

        MOV     BYTE PTR [BX],0        ;
        INC     BX                     ;
        MOV     BYTE PTR [BX],81H      ;
```

```
        CALL    DCON1                       ;
BSFL:   CALL    TDATA                       ;
BFENT:  CALL    KPREG                       ;
        MOV     BP,BLUE                     ;
        MOV     ES,BP                       ;
        CMP     AL,ES:[BX]                  ;
        MOV     BP,GREEN                    ;
        MOV     ES,BP                       ;
        CALL    KPREG                       ;
        JE      $+5                         ;
        JMP     TOGPR                       ;
        CALL    INC8F                       ;
        JMP     SHORT BSFL                  ;
```
データがB面と同じ場合の圧縮処理

```
INC1F:  MOV     AL,0FFH                     ;
        MOV     CS:KIMAX,AL                 ;
        JMP     SHORT INCM1                 ;
INC17:  MOV     AL,7FH                      ;
        MOV     CS:KIMAX,AL                 ;
        JMP     SHORT INCM1                 ;
INC8F:  MOV     AL,0FFH                     ;
        MOV     CS:KIMAX,AL                 ;
        MOV     AL,7FH                      ;
        JMP     SHORT INCM1                 ;
INC13:  MOV     AL,3FH                      ;
        JMP     SHORT INCM                  ;
INC8B:  MOV     AL,0BFH                     ;
        JMP     SHORT INCM                  ;
INCCF:  MOV     AL,0FFH                     ;
        JMP     SHORT INCM                  ;
INC8A:  MOV     AL,0AFH                     ;
        MOV     CS:KIMAX,AL                 ;
        MOV     AL,2FH                      ;
        JMP     SHORT INCM1                 ;
INC45:  MOV     AL,5FH                      ;
        MOV     CS:KIMAX,AL                 ;
        MOV     AL,1FH                      ;
        JMP     SHORT INCM1                 ;
INCBB:  MOV     AL,0BFH                     ;
        JMP     SHORT ICOF                  ;
INC66:  MOV     AL,6FH                      ;
        JMP     SHORT ICOF                  ;
INC77:  MOV     AL,7FH                      ;
ICOF:   MOV     CS:KIMAX,AL                 ;
        MOV     AL,0FH                      ;
        JMP     SHORT INCM1                 ;
INCM:   MOV     CS:KIMAX,AL                 ;
        MOV     AL,3FH                      ;
INCM1:  MOV     CS:KBBSRT,AL                ;
        INC     BYTE PTR [BX]               ;
        MOV     AL,CS:DCONT                 ;
        CMP     AL,2                        ;
        JE      IHLBB                       ;
        MOV     AL,CS:KIMAX                 ;
        CMP     AL,[BX]                     ;
        JE      $+3                         ;
        RET                                 ;
        MOV     AL,CS:KBBSRT                ;
        INC     BX                          ;
        MOV     BYTE PTR [BX],0             ;
        INC     BX                          ;
        MOV     BYTE PTR [BX],AL            ;
        MOV     AL,2                        ;
```
圧縮に合わせたカウント・アップ

```
        MOV     CS:DCONT,AL             ;
        RET                             ;
IHLBB:  XOR     AL,AL                   ;
        CMP     AL,[BX]                 ;
        JE      $+3                     ;
        RET                             ;
        DEC     BX                      ;
        INC     BYTE PTR [BX]           ;
        INC     BX                      ;
        RET                             ;

KIMAX   DB      0FFH                    ;
KBBSRT  DB      0                       ;
KBROCK  DB      0                       ;
KRROCK  DB      0                       ;
KXXOCK  DB      0                       ; ワークエリア
KEQG0   DB      0                       ;
KEQG1   DB      0                       ;
KEQG2   DB      0                       ;
KEQG3   DB      0                       ;
KEQG4   DB      0                       ;

TDAT1:  MOV     AL,1                    ; 圧縮ルーチンで2回目のデータを得る
        MOV     CS:DCONT,AL             ;
TDATA:  CALL    KPREG                   ;
        TEST    CL,01H                  ;
        JE      ADDDE                   ;
        DEC     DI                      ;
        JE      TOADD                   ;
        SUB     BX,DX                   ;
        JMP     SHORT ADDD1             ;
ADDDE:  DEC     DI                      ; 1バイトの画面データを得る
        JE      TOADD                   ;
        ADD     BX,DX                   ;
ADDD1:  MOV     AL,ES:[BX]              ;
        OR      AL,AL                   ;
        CALL    KPREG                   ;
        JNE     $+3                     ;
        RET                             ;
        MOV     CS:KDTCK0,AL            ;
        RET                             ;

TOADD:  INC     CL                      ;
        MOV     AL,CS:XSIZE             ;
        CMP     AL,CL                   ;
        JE      TDEND                   ;
        INC     BX                      ;
        MOV     DI,CS:YSIZE             ; 次の縦列に入った場合の処理
        MOV     AL,ES:[BX]              ; (キャリー・フラグを立てて戻る)
        OR      AL,AL                   ;
        JE      TOAD1                   ;
        MOV     CS:KDTCK0,AL            ;
TOAD1:  CALL    KPREG                   ;
        STC                             ;
        RET                             ;
TDEND:  CALL    KPREG                   ;
        POP     AX                      ;
        MOV     AL,CS:DCONT             ;
        OR      AL,AL                   ;
        JE      TDE1                    ;
        INC     BX                      ;
TDE1:   CMP     BX,DTMAX                ;
        JB      CKEXT                   ;
```

```
MOVER:  MOV     CS:OVERS,1              ;
        MOV     BX,DTMAX-4              ;
        MOV     BYTE PTR [BX],1         ;  最終データ時の処理
        INC     BX                      ;  「POP AX」はSP合わせのダミー
        MOV     BYTE PTR [BX],0         ;  圧縮データ・メモリチェック
        INC     BX                      ;
        MOV     BYTE PTR [BX],0FFH      ;
        INC     BX                      ;
        MOV     BYTE PTR [BX],0FFH      ;
        INC     BX                      ;
CKEXT:  XCHG    DX,BX                   ;
        MOV     AL,CS:KDTCK0            ;
        OR      AL,AL                   ;
        JE      EXRET                   ;
        MOV     BP,CS:KEXIST            ;
        MOV     CS:[BP+0],DX            ;
        STC                             ;
EXRET:  RET                             ;

KEXIST  DW      OFFSET ATOP             ;
KDTCK0  DB      0                       ;
DATAT1  DW      OFFSET DTYPA1           ;  ワークエリア
DATAT2  DW      OFFSET DTYPA2           ;
DATAT3  DW      OFFSET DTYPA3           ;

OPEN3:  ASSUME  DS:DATSEG1              ;
        MOV     AX,DATSEG1              ;
        MOV     DS,AX                   ;
        MOV     SI,CS:DDTOP             ;
        MOV     AL,[SI]                 ;
        ROR     AL,1                    ;
        MOV     BX,OFFSET DTYPB1        ;
        ROR     AL,1                    ;
        JNB     DTPS1                   ;
        MOV     BX,OFFSET DTYPA1        ;
DTPS1:  MOV     CS:DATAT1,BX            ;
        ROR     AL,1                    ;  プレーン別のデータ・タイプ(A/B)により
        MOV     BX,OFFSET DTYPB2        ;  ワークエリアを書き換える
        ROR     AL,1                    ;
        JNB     DTPS2                   ;
        MOV     BX,OFFSET DTYPA2        ;
DTPS2:  MOV     CS:DATAT2,BX            ;
        ROR     AL,1                    ;
        MOV     BX,OFFSET DTYPB3        ;
        ROR     AL,1                    ;
        JNB     DTPS3                   ;
        MOV     BX,OFFSET DTYPA3        ;
DTPS3:  MOV     CS:DATAT3,BX            ;

        XOR     AL,AL                   ;
        MOV     CS:KSHIF1,AL            ;
        MOV     CS:KSHIF2,AL            ;
        MOV     CS:KSHIF3,AL            ;
        MOV     BX,CS:XYPOS             ;
        MOV     DX,80                   ;
        MOV     CL,[SI+1]               ;
        MOV     DI,[SI+2]               ;
        MOV     CS:MAINL1,BX            ;  プログラムの初期化
        MOV     CH,0                    ;
        MOV     CS:MAINL2,CX            ;
        MOV     CS:MAINL3,DI            ;
        CALL    KPREG                   ;
        MOV     BX,CS:DDTOP             ;
```

```
        ADD     BX,DSTART               ;
        MOV     CX,0                    ;
        CALL    KPREG                   ;
        MOV     AL,[SI+0]               ;
        MOV     CS:KBRGDT,AL            ;

        MOV     AX,BLUE                 ;
        MOV     BX,OFFSET KSHIF1        ; B面の展開/表示
        MOV     DX,OFFSET GGDA1         ;
        CALL    MAINL                   ;

        MOV     AX,RED                  ;
        MOV     BX,OFFSET KSHIF2        ; R面の展開/表示
        MOV     DX,OFFSET GGDA2         ;
        CALL    MAINL                   ;

        MOV     AX,GREEN                ;
        MOV     BX,OFFSET KSHIF3        ; G面の展開/表示
        MOV     DX,OFFSET GGDA3         ;
        CALL    MAINL                   ;

        MOV     AL,[SI]                 ;
        MOV     BX,OFFSET DTYPB1        ;
        AND     AL,80H                  ;
        JE      DTPS4                   ; ワークエリア初期化
        MOV     BX,OFFSET DTYPA1        ;
DTPS4:  MOV     CS:DATAT1,BX            ;
        XOR     AL,AL                   ;
        MOV     CS:KSHIF1,AL            ;

        MOV     AX,ITSTY                ;
        MOV     BX,OFFSET KSHIF1        ; I面の展開/表示
        MOV     DX,OFFSET GGDA1         ;
        CALL    MAINL                   ;
        RET

KBRGDT  DB      0                       ;
KLPON1  DW      OFFSET KSHIF1           ;
KLPON2  DW      OFFSET GGDA1            ;
KCPON1  DW      BLUE                    ; ワークエリア
KCPON2  DW      BLUE                    ;
KCPON3  DW      BLUE                    ;
MAINL1  DW      0                       ;
MAINL2  DW      0                       ;
MAINL3  DW      0                       ;

MAINL:  MOV     ES,AX                   ;
        MOV     CS:KCPON1,AX            ; 展開用セグメント値のセット及び保存
        MOV     CS:KCPON2,AX            ;
        MOV     CS:KCPON3,AX            ;

        MOV     AL,CS:KBRGDT            ;
        ROR     AL,1                    ; 無データならNDISPへ
        JB      $+5                     ;
        JMP     NDISP                   ;

        ROR     AL,1                    ;
        MOV     CS:KBRGDT,AL            ;
        MOV     CS:KLPON1,BX            ;
        MOV     CS:KLPON2,DX            ; プログラム初期化
        MOV     BX,CS:MAINL1            ;
        MOV     CX,CS:MAINL2            ;
        MOV     DI,CS:MAINL3            ;
```

```
        MOV     DX,80                   ;
LOLOP:  PUSH    CX                      ;
        PUSH    DI                      ;
        MOV     BP,CS:KLPON1            ;
        MOV     AL,CS:[BP+0]            ;
        AND     AL,00000011B            ;
        JE      LOLLP                   ;
        CALL    KPREG                   ;
        DEC     AL                      ;
        JE      LPSF1                   ;
        DEC     AL                      ;
        JE      LPSF2                   ;
        MOV     AL,DH                   ;
        MOV     DH,DL                   ;
        MOV     DL,AL                   ;
        JMP     SHORT JLPON2            ;
LPSF2:  ROR     DL,1                    ;
        ROR     DH,1                    ;
LPSF1:  ROR     DL,1                    ;
        ROR     DH,1                    ;
        CALL    KPREG                   ;
LOLLP:  CALL    KPREG                   ;
JLPON2: CALL    CS:KLPON2               ;
        CALL    KPREG                   ;
        MOV     ES:[BX],AL              ;
        ADD     BX,DX                   ;
        DEC     DI                      ;
        JNE     LOLLP                   ;
        INC     BX                      ;
        POP     DI                      ;
        POP     CX                      ;
        DEC     CX                      ;
        JNE     $+3                     ;
        RET                             ;
        INC     DH                      ;
        JNE     LPMHL                   ;
        MOV     DX,80                   ;
        ADD     BX,DX                   ;
        JMP     SHORT LOLOP             ;
LPMHL:  MOV     DX,-80                  ;
        ADD     BX,DX                   ;
        JMP     SHORT LOLOP             ;

GGDA1:  OR      CX,CX                   ;
        JE      NEWDT1                  ;
        DEC     CX                      ;
        MOV     AL,CS:KSHIF1            ;
SHIF12: OR      AL,AL                   ;
        JE      DAFIX1                  ;
        DEC     AL                      ;
        JE      SFT11                   ;
        DEC     AL                      ;
        JE      SFT12                   ;
        MOV     AL,DH                   ;
        MOV     DH,DL                   ;
        MOV     DL,AL                   ;
        JMP     SHORT GDA1RT            ;
SFT12:  ROR     DL,1                    ;
SFT11:  ROR     DL,1                    ;
DAFIX1: MOV     AL,DL                   ;
GDA1RT: RET                             ;
```

LOLOP〜LPMHL部：展開したデータを画面に表示する
(縦列変更時は必要に応じDXをローテートする)

GGDA1〜GDA1RT部：B面用データ展開プログラム
CX＝圧縮データ連続数
DH＝前回のデータ
DL＝今回のデータ

```
NEWDT1: MOV     AL,[BX]         ;
        INC     BX              ;
        MOV     DL,AL           ;
        ROR     AL,1            ;
        ROR     AL,1            ;
        ROR     AL,1            ;
        ROR     AL,1            ;
        CMP     AL,DL           ;
        MOV     AL,DL           ;
        JE      $+3             ;
        RET                     ;
        INC     AL              ;
        JE      GSDTF1          ;
        DEC     AL              ;
        JE      GSDT01          ;
        CMP     AL,55H          ;
        JE      GSD5A           ;
        CMP     AL,0AAH         ;
        JE      GSD5A           ;
        JMP     CS:DATAT1       ;
DTYPB1: CMP     AL,33H          ;
        JE      DTYPA1          ; 新しいデータに圧縮がかかっているか
        CMP     AL,66H          ; どうかをチェックする
        JE      DTYPA1          ;
        CMP     AL,99H          ;
        JE      DTYPA1          ;
        CMP     AL,0CCH         ;
        JE      $+3             ;
        RET                     ;
DTYPA1: MOV     AL,2            ;
GSD5X1: MOV     CL,[BX]         ;
        INC     BX              ;
        TEST    CL,80H          ;
        JE      CONT011         ;
        AND     CL,07FH         ;
CONT01: XOR     AL,AL           ;
CONT011:MOV     CS:KSHIF1,AL    ;
        MOV     AL,7FH          ;
STDCK0: CMP     AL,CL           ;
        JNE     GOOK1           ;
        MOV     CH,[BX]         ;
        INC     BX              ;
        MOV     CL,[BX]         ;
        INC     BX              ;
GOOK1:  DEC     CX              ;
        MOV     AL,DL           ;
        RET                     ;

GSD5A:  MOV     AL,1            ; 圧縮サインが55H／AAHの場合の展開
        JMP     SHORT GSD5X1    ;

GSDT01: MOV     AL,[BX]         ;
        INC     BX              ;
        OR      AL,AL           ;
        JE      GS0001          ; 先頭圧縮サインが00Hの場合の展開
        MOV     CL,AL           ;
        JMP     SHORT GS00B1    ;

GS0001: MOV     DL,[BX]         ;
        INC     BX              ;
        MOV     CL,[BX]         ;
        INC     BX              ; 圧縮サインが00H,00Hの場合の展開
GS00B1: XOR     AL,AL           ;
```

```
        MOV     CS:KSHIF1,AL                    ;
        DEC     AL                              ;
        JMP     SHORT STDCK0                    ;

GSDTF1: MOV     CL,[BX]                         ;
        INC     BX                              ; 先頭圧縮サインがFFHの場合の展開
        JMP     SHORT GS00B1                    ;

GGDA2:  OR      CX,CX                           ;
        JE      NEWDT2                          ;
        DEC     CX                              ;
        MOV     AL,CS:KSHIF2                    ; R面データ展開プログラム
        TEST    AL,20H                          ; (B面参照でなければSHIF12へ)
        JNE     $+5                             ;
        JMP     SHIF12                          ;

BMENS2: MOV     BP,BLUE                         ;
        MOV     ES,BP                           ;
BMENS22:                                        ;
        TEST    AL,80H                          ;
        JNE     RRB0                            ;
        AND     AL,00000011B                    ;
        JE      RRDB0                           ;
        DEC     AL                              ;
        JE      RRDB1                           ;
        ROR     DH,1                            ;
RRDB1:  ROR     DH,1                            ; B面を参照してデータを得る場合
RRDB0:  CALL    KPREG                           ;
        MOV     AL,ES:[BX]                      ;
        CALL    KPREG                           ;
        AND     AL,DH                           ;
        MOV     ES,CS:KCPON1                    ;
        RET                                     ;
RRB0:   CALL    KPREG                           ;
        MOV     AL,ES:[BX]                      ;
        CALL    KPREG                           ;
        MOV     ES,CS:KCPON2                    ;
        RET

NEWDT2: MOV     AL,[BX]                         ;
        INC     BX                              ;
        MOV     DL,AL                           ;
        ROR     AL,1                            ;
        ROR     AL,1                            ;
        ROR     AL,1                            ;
        ROR     AL,1                            ;
        CMP     AL,DL                           ;
        MOV     AL,DL                           ;
        JE      $+3                             ;
        RET                                     ;
        INC     AL                              ;
        JNE     $+5                             ;
        JMP     GSDTF2                          ;
        DEC     AL                              ;
        JE      GSDT02                          ;
        CMP     AL,55H                          ;
        JNE     $+5                             ;
        JMP     GSD5A2                          ;
        CMP     AL,0AAH                         ;
        JNE     $+5                             ;
        JMP     GSD5A2                          ;
        JMP     CS:DATAT2                       ;
```

```
DTYPB2: CMP     AL,33H          ;
        JE      DTYPA2          ;
        CMP     AL,66H          ;
        JE      DTYPA2          ;
        CMP     AL,99H          ;  新しいデータに圧縮がかかっているかどう
        JE      DTYPA2          ;  をチェックする
        CMP     AL,0CCH         ;
        JE      $+3             ;
        RET                     ;
DTYPA2: MOV     CL,[BX]         ;
        MOV     AL,2            ;
        INC     BX              ;
        TEST    CL,80H          ;
        JE      CONT02          ;
        AND     CL,07FH         ;
        TEST    CL,40H          ;
        JNE     $+5             ;
        JMP     BAND00          ;
        AND     CL,0BFH         ;
        MOV     DH,DL           ;
        CALL    KPREG           ;
        MOV     BP,BLUE         ;
        MOV     ES,BP           ;
        MOV     AL,ES:[BX]      ;
        CALL    KPREG           ;
        AND     AL,DH           ;
        MOV     DL,AL           ;
        MOV     AL,00100010B    ;
        JMP     CONT12

GSDT02: MOV     AL,[BX]         ;
        INC     BX              ;
        OR      AL,AL           ;
        JE      GS0002          ;
        MOV     CL,AL           ;
        AND     AL,10000000B    ;
        JE      CONT02          ;
        AND     CL,07FH         ;
        MOV     BP,BLUE         ;
        MOV     ES,BP           ;
        CALL    KPREG           ;
        MOV     AL,ES:[BX]      ;
        CALL    KPREG           ;  先頭圧縮サインが00Hの場合の展開
        MOV     DL,AL           ;
        MOV     AL,10100000B    ;
CONT02: MOV     CS:KSHIF2,AL    ;
STDC12: MOV     AL,7FH          ;
STDCK1: CMP     AL,CL           ;
        JNE     GOOK11          ;
        MOV     CH,[BX]         ;
        INC     BX              ;
        MOV     CL,[BX]         ;
        INC     BX              ;
GOOK11: DEC     CX              ;
        MOV     AL,DL           ;
        MOV     ES,CS:KCPON3    ;
        RET                     ;

KSHIF1  DB      0               ;
KSHIF2  DB      0               ;  ワークエリア
KSHIF3  DB      0               ;

GSDTF2: MOV     CL,[BX]         ;
```

```
        INC     BX                  ; 先頭圧縮サインがFFHの場合の展開
        JMP     SHORT GS00R2        ;

GS0002: MOV     DL,[BX]             ;
        INC     BX                  ;
        MOV     CL,[BX]             ;
        INC     BX                  ; 圧縮サインが00H,00Hの場合の展開
GS00R2: XOR     AL,AL               ;
        MOV     CS:KSHIF2,AL        ;
        DEC     AL                  ;
        JMP     SHORT STDCK1        ;

GSD5A2: MOV     CL,[BX]             ;
        MOV     AL,1                ;
        INC     BX                  ;
        TEST    CL,80H              ;
        JE      CONT02              ;
        AND     CL,07FH             ;
        TEST    CL,40H              ;
        JNE     BMA52               ;
BAND00: TEST    CL,20H              ;
        JE      RRRR0               ;
        TEST    CL,10H              ;
        JE      RRRR0               ;
        AND     CL,0DFH             ;
        AND     CL,0EFH             ;
        MOV     DH,DL               ;
        CALL    KPREG               ;
        MOV     BP,BLUE             ;
        MOV     ES,BP               ; 圧縮サインが55H,AAHの場合の展開
        MOV     AL,ES:[BX]          ;
        CALL    KPREG               ;
        AND     AL,DH               ;
        MOV     DL,AL               ;
        MOV     AL,00100000B        ;
        MOV     CS:KSHIF2,AL        ;
        MOV     AL,0FH              ;
        JMP     SHORT STDCK1        ;

BMA52:  AND     CL,0BFH             ;
        MOV     DH,DL               ;
        CALL    KPREG               ;
        MOV     BP,BLUE             ;
        MOV     ES,BP               ;
        MOV     AL,ES:[BX]          ;
        CALL    KPREG               ;
        AND     AL,DH               ;
        MOV     DL,AL               ;
        MOV     AL,00100001B        ;
CONT12: MOV     CS:KSHIF2,AL        ;
        MOV     AL,3FH              ;
        JMP     STDCK1              ;

RRRR0:  XOR     AL,AL               ; 上位／下位とも同じデータが連続している
        MOV     CS:KSHIF2,AL        ; 圧縮の展開
        MOV     AL,2FH              ;
        JMP     STDCK1              ;

GGDA3:  OR      CX,CX               ;
        JE      NEWDT3              ;
        DEC     CX                  ;
        MOV     AL,CS:KSHIF3        ; G面用データ展開プログラム
        TEST    AL,20H              ;
```

```
        JE      $+5                     ;
        JMP     BMENS2                  ;
        TEST    AL,10H                  ;
        JNE     $+5                     ;
        JMP     SHIF12                  ;
        MOV     BP,RED                  ;
        MOV     ES,BP                   ;
        JMP     BMENS22                 ;

NEWDT3: MOV     AL,[BX]                 ;
        INC     BX                      ;
        MOV     DL,AL                   ;
        ROR     AL,1                    ;
        ROR     AL,1                    ;
        ROR     AL,1                    ;
        ROR     AL,1                    ;
        CMP     AL,DL                   ;
        MOV     AL,DL                   ;
        JE      $+3                     ;
        RET                             ;
        INC     AL                      ;
        JNE     $+5                     ;
        JMP     GSDTF3                  ;
        DEC     AL                      ;
        JE      GSDT03                  ;
        CMP     AL,55H                  ;
        JNE     $+5                     ;
        JMP     GSD5A3                  ;
        CMP     AL,0AAH                 ;
        JNE     $+5                     ;
        JMP     GSD5A3                  ;
        JMP     CS:DATAT3               ;  新しいデータに圧縮がかかっているか
                                        ;  どうかをチェック
DTYPB3: CMP     AL,33H                  ;
        JE      DTYPA3                  ;
        CMP     AL,66H                  ;
        JE      DTYPA3                  ;
        CMP     AL,99H                  ;
        JE      DTYPA3                  ;
        CMP     AL,0CCH                 ;
        JE      $+3                     ;
        RET                             ;
DTYPA3: MOV     CL,[BX]                 ;
        INC     BX                      ;
        TEST    CL,80H                  ;
        JNE     GX081                   ;
        MOV     AL,2                    ;
        JMP     GSDXX0

GX081:  MOV     DH,DL                   ;
        AND     CL,07FH                 ;
        TEST    CL,40H                  ;
        JNE     RMAXX                   ;
        CALL    KPREG                   ;
        MOV     BP,BLUE                 ;  データが「B面 AND DH」の連続の場合の
        MOV     ES,BP                   ;  展開（DHは2ローテートする）
        MOV     AL,ES:[BX]              ;
        CALL    KPREG                   ;
        AND     AL,DH                   ;
        MOV     DL,AL                   ;
        MOV     AL,00100010B            ;
        JMP     CONT13
```

```
RMAXX:  AND     CL,0BFH               ;
        CALL    KPREG                 ;
        MOV     BP,RED                ;
        MOV     ES,BP                 ;
        MOV     AL,ES:[BX]            ; データが「R面 AND DH」の連続の場合の
        CALL    KPREG                 ; 展開（DHは2ローテートする）
        AND     AL,DH                 ;
        MOV     DL,AL                 ;
        MOV     AL,00010010B          ;
        JMP     CONT13

GSDT03: MOV     AL,[BX]               ;
        INC     BX                    ;
        OR      AL,AL                 ;
        JE      GS0003                ;
        MOV     CL,AL                 ;
        AND     AL,10000000B          ;
        JE      CONT03                ;
        AND     CL,07FH               ; 先頭圧縮サインが00Hの場合の展開
        CALL    KPREG                 ;
        MOV     BP,BLUE               ;
        MOV     ES,BP                 ;
        MOV     AL,ES:[BX]            ;
        CALL    KPREG                 ;
        MOV     DL,AL                 ;
        MOV     AL,10100000B          ;
CONT03: MOV     CS:KSHIF3,AL          ;
        JMP     STDC12

GS0003: MOV     DL,[BX]               ;
        INC     BX                    ;
        MOV     CL,[BX]               ;
        INC     BX                    ; データが交互連続の場合の展開
        XOR     AL,AL                 ;
        MOV     CS:KSHIF3,AL          ;
        DEC     AL                    ;
        JMP     STDCK1

GSDTF3: MOV     CL,[BX]               ;
        INC     BX                    ;
        XOR     AL,AL                 ;
        TEST    CL,80H                ;
        JE      CONT03                ;
        AND     CL,07FH               ;
        CALL    KPREG                 ; 先頭圧縮サインがFFHの場合の展開
        MOV     BP,RED                ;
        MOV     ES,BP                 ;
        MOV     AL,ES:[BX]            ;
        CALL    KPREG                 ;
        MOV     DL,AL                 ;
        MOV     AL,10010000B          ;
        JMP     SHORT CONT03

GSD5A3: MOV     CL,[BX]               ;
        INC     BX                    ;
        TEST    CL,80H                ;
        JNE     G5081                 ;
        MOV     AL,1                  ;
GSDXX0: TEST    CL,40H                ; 連続の数え方により分岐
        JE      CONT13                ;
        AND     CL,0BFH               ;
        TEST    CL,20H                ;
        JE      GRRR0
```

```
         MOV     DH,DL               ;
         AND     CL,0DFH             ;
         TEST    CL,10H              ;
         JNE     RMRR0               ;

         CALL    KPREG               ;
         MOV     BP,BLUE             ;
         MOV     ES,BP               ;
         MOV     AL,ES:[BX]          ;
         CALL    KPREG               ;  データが「B面 AND DH」の連続
         AND     AL,DH               ;  の場合の展開
         MOV     DL,AL               ;
         MOV     AL,00100000B        ;
CON00F:  MOV     CS:KSHIF3,AL        ;
         MOV     AL,0FH              ;
         JMP     STDCK1

RMRR0:   AND     CL,0EFH             ;
         CALL    KPREG               ;
         MOV     BP,RED              ;
         MOV     ES,BP               ;
         MOV     AL,ES:[BX]          ;  データが「R面 AND DH」の連続
         CALL    KPREG               ;  の場合の展開
         AND     AL,DH               ;
         MOV     DL,AL               ;
         MOV     AL,00010000B        ;
         JMP     SHORT CON00F

GRRR0:   XOR     AL,AL               ;
         MOV     CS:KSHIF3,AL        ;  上位／下位が同じデータ（ローテートする）
         MOV     AL,1FH              ;  の場合の展開
         JMP     STDCK1              ;
G5081:   MOV     DH,DL               ;
         AND     CL,07FH             ;
         TEST    CL,40H              ;
         JNE     RMA55               ;
         CALL    KPREG               ;
         MOV     BP,BLUE             ;
         MOV     ES,BP               ;  データが「B面 AND DH」の連続
         MOV     AL,ES:[BX]          ;  の場合の展開（DHは1ローテートする）
         CALL    KPREG               ;
         AND     AL,DH               ;
         MOV     DL,AL               ;
         MOV     AL,00100001B        ;
CONT13:  MOV     CS:KSHIF3,AL        ;
         MOV     AL,3FH              ;
         JMP     STDCK1

RMA55:   AND     CL,0BFH             ;
         CALL    KPREG               ;
         MOV     BP,RED              ;
         MOV     ES,BP               ;
         MOV     AL,ES:[BX]          ;  データが「R面 AND DH」の連続
         CALL    KPREG               ;  の場合の展開（DHは1ローテートする）
         AND     AL,DH               ;
         MOV     DL,AL               ;
         MOV     AL,00010001B        ;
         JMP     SHORT CONT13

NDISP:   ROR     AL,1                ;
         MOV     CS:KBRGDT,AL        ;
         XOR     AX,AX               ;
         MOV     DX,AX
```

```
        MOV     DL,[SI+1]                       ;
        MOV     BX,[SI+2]                       ;
        MOV     BP,CS:XYPOS                     ;
NDILO:  MOV     CX,DX                           ; 無データのプレーンを指定サイズでクリアする
        MOV     DI,BP                           ;
        REP     STOSB                           ;
        ADD     BP,80                           ;
        DEC     BX                              ;
        JNE     NDILO                           ;
        RET                                     ;

KEPBX   DW      0                               ;
KEPCX   DW      0                               ; レジスタ保存用ワークエリア
KEPDX   DW      0                               ;

KPREG:  XCHG    BX,CS:KEPBX                     ;
        XCHG    CX,CS:KEPCX                     ; レジスタ保存用
        XCHG    DX,CS:KEPDX                     ;
        RET

CODE    ENDS

DATSEG1 SEGMENT PUBLIC                          ;データ保存用セグメントの定義
DTOP    DB      0FFFFH  DUP(?)
DATSEG1 ENDS

        END
```

5－3　グラフィックス開発用支援ツール

「顔じゃないよ、心だよ……」と言いながらも、本心は美男美女に憧れているのが人間というものです。どんなにゲームのアイデアやプログラミング・テクニックが優れていても、それだけではゲームとしての魅力を満喫することはできません。ゲームの顔、それはやはり画面を構成するグラフィックスです。

これまでに紹介してきた種々のテクニックも、画面次第で価値は大きく変化します。いかにしてこの価値を高めるか、グラフィック・ツールの存在を無視してゲームを語ることはできないでしょう。これから紹介する各種ツールは、プロ用として私が開発したものを一般用にリメイクしたものです。めったに使われることのない特殊機能はカットしていますが、その分使い勝手を向上させてあります。市販のゲームと同等のグラフィックスを作成し、しかも本書で紹介したテクニックに合ったデータを得ることができます。大いに活用してください。

プログラムは、大型グラフィックス用の『PMAN98』、キャラクタ作成用の『PTER98』、スキャナコントロール用の『SCAN98』とに分かれていますが、これらは実行ファイルとしてセーブしてあります（PMAN98.EXE/PTER98.EXE/SCAN98.EXE）。これらは単独でも、あるいはメニュー（GMENU.COM）からでも実行できるようになっています。では、さっそく実行してみましょう。

```
A>GMENU⏎
```

メニュー画面(**写真 1**)が現れたら、希望の番号を入力してください。矢印(↑↓)キーで選択し、リターン・キーで実行することも可能です。

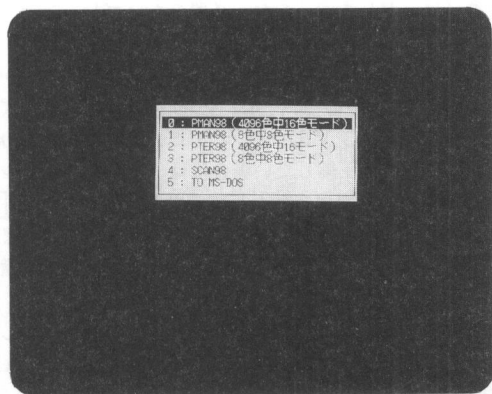



写真 1　メニュー画面

【PMAN98】を選択すると……

画面は『PMAN98』のメニューに変わります。ここで HELP キーを押すと、このメニュー画面で実行できる内容が示されます(**写真 2**)。それぞれの内容は次のとおりです。コマンドからの回避は、特に指定のない限り ESC キーでできます。

A～D　：セーブ／ロードなどのファイルを扱う場合のドライブ指定です。
E　：グラフィックをエディットするモードへ移ります。

```
== PMAN98 == (C)1991 by T.Hidaka & M.Aoyama ver.1.0

[A]
------------------------------------
 A~D     ･･･ Drive  A~D
 E       ･･･ Edit
 F       ･･･ Dir *.*
 K       ･･･ Kill file
 L       ･･･ Load & Display
 SHIFT+L ･･･ Load (plane 0~3)
 M       ･･･ To MS-DOS
 P       ･･･ Press data
 S       ･･･ Save File
 SHIFT+S ･･･ Save (plane 0~3)
 V       ･･･ Load *.IM?
 Z       ･･･ Load *.ST4
 *       ･･･ Hard copy
 /       ･･･ Hard copy (1/2)
 Space   ･･･ Text/Graphic
 2468    ･･･ Graphic move
 Esc     ･･･ End Pman98
------------------------------------
[A] █
```

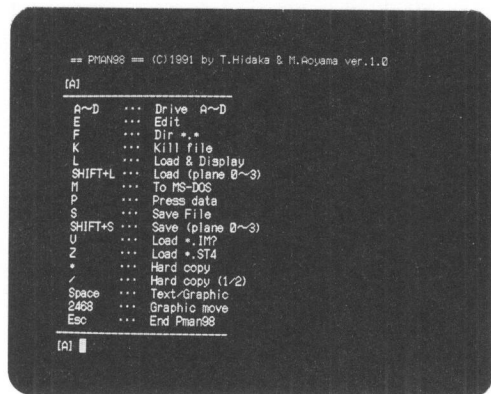



写真 2 『PMAN98』のメニュー画面

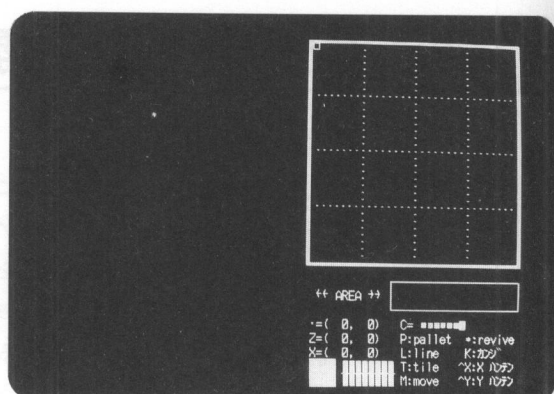



写真 3 『PMAN98』のエディット画面

F　　：指定されたドライブの全ファイル名を表示します。

K　　：指定されたドライブにあるファイルを削除します。ファイル名を聞いてきますから、間違えないように入力してください。

L　　：圧縮されたグラフィック・データをロードします。ファイルを一覧表示しますから、カーソルキー（↑↓→←）で選択してください。ファイルの決定はリターンキーです。なお、マニュアル入力も可能となっています。また、ワイルド・カードを指定した場合には、ファイルをワイルド・カードに従って、表示するように設定してあります。
リターンキーで決定したファイルのデータ形式が違うと「NOT DATA FILE!! 」と表示されます。ただし、圧縮されたデータかどうかを完全に判定することはできませんので、自分なりにファイル名に工夫を凝らして間違えないようにしてください。

SHIFT＋L　：Lコマンドと操作方法は同じですが、この場合のデータは、プレーン別のベタデータを順番にロードします。なお、必要のないプレーンデータは、 ESC キーでスキップすることができます。

P　　：グラフィック画面のデータを圧縮します。画面には圧縮をかけるエリアを指定する枠が現れますので、テンキー（2,4,6,8）にて調整してください。この時、左上は常に画面左上（G.VRAMのオフセットアドレス 0000Hの位置）に固定されています。中央部を圧縮したい場合には、一旦コマンドレベルに戻り、テンキーの（2,4,6,8）にて画面を動かしてください。このとき SPACE キーでグラフィック画面表示状態にしておくと、より見やすくなります。データ圧縮が完了すると、データアドレスと圧縮率が示されます。なお、横のサイズは8ドット単位となっています。

S　　：圧縮されたデータをセーブしますから、ファイル名を入力してください。圧縮されたデータがない場合には、「SAVE DATA NOT READY!!」と表示され、セーブできないようになっています。

SHIFT＋S　：Sコマンドと操作方法は同じですが、データは圧縮をかけずにプレーン別にベタでセーブしていきます。必要のないプレーンは、 ESC キーでスキップすることができます。

V　　：Lコマンドと操作方法は同じですが、『ダ・ビンチ』のデータ・ファイルをロード／展開するためのコマンドです。

Z　　：Lコマンドと操作方法は同じですが、『Z'S STAFF』シリーズの画面データ（640×400：拡張子＝.ST4）をロード／展開するためのコマンドです。

（※『Z'S STAFF』は（株）ツァイトの登録商標です）

＊ ：グラフィック画面をカラーハードコピーします。カラープリンタのスイッチを入れ、準備を整えておいてください。モノクロのプリンタの場合は、白以外はすべて黒になります。エリアの指定はデータ圧縮の時と同じです。

／ ：グラフィック画面を1280ドットモードでカラーハードコピーします。つまり、全体が 1/2に縮小されてコピーされるわけですが、このモードでは縦ラインが1ラインずつスキップされたようになります。また、エリア指定の縦は16ライン単位となります。

SPACE ：スペースキーを押すたびに、テキスト画面／グラフィック画面が交互に切り換わります。

[2],[4],[6],[8] ：グラフィック画面を上下左右に動かします。なお、[CTRL] キーを押しながら移動させるとドット単位に動かすことができます。

ESC ：GMENU.COM によって起動した場合にはメニュー画面（写真 1）へ戻ります。また、MS-DOSのコマンドラインから起動した場合には、MS-DOSへ戻ります。

M ：『PMAN98』の実行を終了し、MS-DOSへダイレクトに戻ります。

では、[E] キーを押して画面エディットモード（写真 3）へ入ってみましょう。画面右に、グラフィックスを拡大したものが表示され、その下にはエディット中の状態や使用できる主要なコマンドを示す欄があります。コマンドによっては確認や質問をしてくるものがありますが、それらは枠で囲まれたメッセージ欄に現れます。カーソルの移動はテンキーによって行い、現在のドット位置は「・＝(0,0) 」の部分に示されます。[SHIFT] キーを押すと高速に移動できます。

［エディタ部におけるコマンド］

C ：色の指定。右下部に表示されているカラーテーブル上のカーソルを[4][6]キーで移動して指定します。色の決定は、リターンキーです。また、カラーテーブルの右端は、下部に示されるタイリングカラーのことです。

Z ：ドット位置の記憶－①

X ：ドット位置の記憶－②

P ：パレットの変更

《4096色中16色モード時》

カーソルキー（↑↓）で変更したいパレット番号を指定します。パレット番号を指定したら、テンキー（[2],[8]）でパレットを構成する色の成分（青、赤、緑）を選択します。さらに、テンキーの（[4],[6]）キーでそのレベル（16段階）を決定します。

《8色中8色モード時》

パレット番号で指定します。

M ：Z－Xで囲まれたグラフィックスを、現在カーソルのある位置に移動（コピー）します。

L ：Z－Xを結ぶラインまたはボックスを描きます。

T ：タイリングカラーによるペイントを行います。通常のペイント、ボックスフィルの他にスーパータイリングという特殊ペイントがあります。これは、Z－Xで囲まれたエリアにある指定の一色（または黒以外の全色）をタイリングカラーで塗るというものです。タイリングカラーの指定は、実際に色を確認しながら作成することができます。

＊ ：拡大表示されている部分に限り、初期状態（そのボックス内に入った時の状態）に戻すことができます。

K ：漢字表示。漢字コード表を見てコード番号で入力してください。

^X ：Z－Xで囲まれた部分を左右反転します。

^Y　：Z－Xで囲まれた部分を上下反転します。

←→　：エディットエリアの変更。右側の隠された部分をエディットしたい場合に使用しますが、ペイントや移動などすべてのコマンドは画面に表示されているエリアに対してのみ行われます。そのため、Z－Xによる指定も連動して動きますから注意してください。

カナ　：［カナ］キーをロックすると、拡大されている部分がボックスで示され、移動もボックス単位となります。

H.CLR：カーソルを左上に移動します。［カナ］キーロックの場合は拡大ボックスを左上に移動させます。

／,.　：［H.CLR］キーと同様に、カーソルあるいは拡大ボックスを、それぞれ右上、左下、右下に移動させます。

ESC　：『PMAN98』のメニューに戻ります。

【PTER98】を選択すると……

画面は『PTER98』のメニューに変わります。ここで［HELP］キーを押すと、このメニュー画面で実行できる内容が示されます(**写真** 4)。それぞれの内容は次のとおりです。

A～D：セーブ／ロードなどのファイルを扱う場合のドライブの指定です。

E　：キャラクタ・パターンをエディットするモードへ移ります。

F　：指定されたドライブの全ファイル名を表示します。

K　：指定されたドライブにあるファイルを削除します。

L　：パターンデータをロードします。全ファイル名を表示しますから、番号で答えてください。なお、ワイルド・カードの指定も可能です。

S　：データをセーブしますから、ファイル名を入力してください。

ESC　：GMENU.COMによって起動した場合にはメニュー画面（写真 1）へ戻ります。また、MS-DOSのコマンドラインから起動した場合にはMS-DOSへ戻ります。

M　：『PTER98』の実行を終了し、MS-DOSへダイレクトに戻ります。

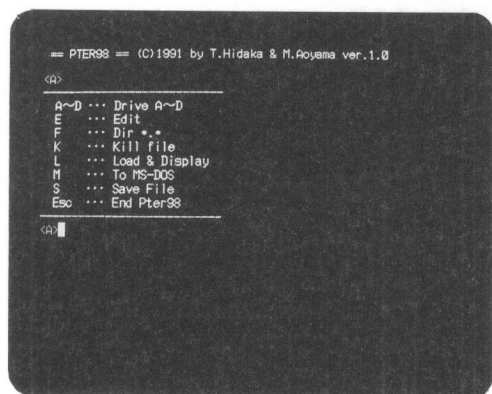



写真 4 『PTER98』のメニュー画面

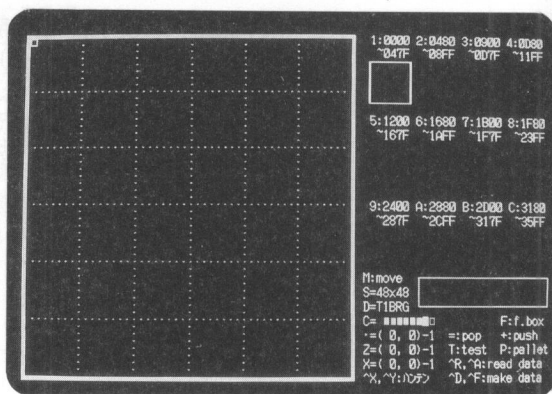



写真 5 『PTER98』のエディット画面

では、［E］キーを押して画面エディットモード(**写真** 5) へ入ってみましょう。画面左に、パターンを拡大したものが表示されます。『PMAN98』とほとんど同じですが、キャラクタ・パターン用ということでいくつかの違いがあります。

［エディタ部におけるコマンド］

S ：キャラクタ・パターンのサイズを示しています。作成したいパターンのサイズに合わせてください。

D ：データのタイプを示しています。通常のBRGIデータと、くり抜きデータを含んだものがあります。 透明部分のビットをゼロとするタイプが「T0BRGI」で、1となるタイプが「T1BRGI」となりますから、プログラムによって使い分けてください。

C ：色の指定。右下部に表示されているカラーテーブル上のカーソルをテンキー(4,6)で移動して指定します。色の決定はリターンキーです。また、カラーテーブルの右端は、下部に示されるタイリングカラーのことです。

Z ：ドット位置の記憶—①

X ：ドット位置の記憶—②

^X ：Z—Xで囲まれた部分を左右反転します。

^Y ：Z—Xで囲まれた部分を上下反転します。

＋ ：エディット中のパターンを記憶します。

＝ ：記憶してあるパターンを再生します。

F ：Z—Xで囲まれた部分をカラー番号で示された色で塗りつぶします。

M ：Z—Xで囲まれたグラフィックスを、現在カーソルのある位置に移動（コピー）します。

T ：パターンのアニメーション・テストを行います。最初にアニメーションの順序を入力し、次に好みのウェイトを入力してください。なお、アニメーション中は、スペースキーで一時停止することができます。

P ：パレットの変更

《4096色中16色モード時》

カーソルキー（↑↓）で変更したいパレット番号を指定します。パレット番号を指定したら、テンキー(2,8) でパレットを構成する色の成分(青、赤、緑) を選択します。さらに、テンキーの(4,6) キーでそのレベル（16段階）を決定します。

《8色中8色モード時》

パレット番号で指定します。

^R ：エディット中のパターン・データをリードし画面上に表示します。

^A ：すべてのパターン・データをリードし画面上に表示します。

^D ：エディット中のパターンをデータ化します。

^F ：すべてのパターンをデータ化します。

カナ ：［カナ］キーをロックすると、エディット・パターンを上下左右に移動させることができます。

H.CLR ：カーソルを左上に移動します。

／,. ：それぞれカーソルを右上、左下、右下に移動します。

ESC ：『PTER98』のメニューに戻ります。

【SCAN98】を選択すると……

画面は『SCAN98』のコントロール画面（写真 6）に変わります。スキャナは種類によって機能が違いますから、必ず機種を合わせてください。主要な部分は機種設定によって選択肢が制限されるようになっています。ただし、スキャナは細かな部分における仕様の違いが結構多いため、プログラム側ですべてについて正確に判断しているわけではありません。そのため、サイズや DPIなどについてはマニュアルでは不可能な設定もできることがあるかもしれませんが、それらは無視されますので注意してください。なお、各機能の設定は DPIの一部を除いてすべてロータリー

式になっています。また、実際にスキャナを使用する場合には、ボーレート等の通信条件を本体とスキャナ側で一致させる必要があります。本ソフトウェアは、下記の条件で動作を確認しておりますので最初に本体／スキャナの設定を行ってください。☆印は、スキャナが本体側に要求している設定です。

＊ RS-232C 通信条件

☆ボーレート　　　：9600ボー
　通信方式　　　　：全二重
　Ｘパラメータ　　：無効
☆ストップビット長：１ビット
☆データビット長　：８ビット
　Ｓパラメータ　　：無効
　ＤＥＬコード　　：有効
☆パリティチェック：なし
　パリティ　　　　：偶数

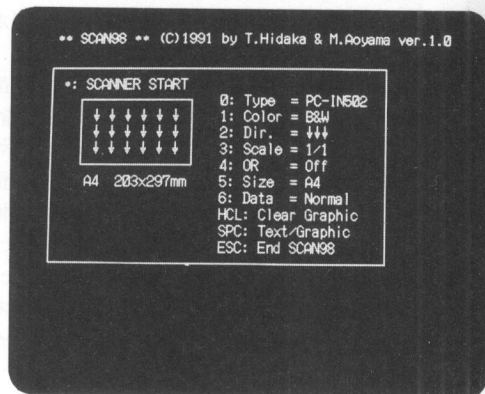



写真 6 『SCAN88』のコントロール画面

[SCAN98におけるコマンド]

0 ：スキャナの機種設定。こちらで登録してあるのは次の各機種ですが、ここに示されたもの以外でもマニュアルに「機能は……に準ずる」とたいてい表示されているはずですから、それに合わせてお使いください。なお、HS40CL／HS60F(オムロン)のように、セーブデータが『Z'S STAFF』シリーズのファイル「＊.ST4」に対応している場合は、『PMAN98』でファイルを直接ロード／展開（Ｚコマンド）することができます。**登録済み機種**:PC-IN502,PC-IN503,PC-IN503H,PC-IN505,PC-IN506,PC-PR406S,PC-PR406HS（以上、日本電気）HS7R,HS7R2,HS10R,HS10R2(以上、オムロン)

1 ：モノクロ、カラー（面順次／線順次）の選択

2 ：画面表示方向の選択（縦／横）

3 ：画面表示の際の縮小率（1/1 ,1/2）

4 ：スキャナ側で読み取ったデータを縮小する場合、間引かれたデータを隣接するデータとORを取って残すかどうかの選択です。ただし、機種によってはこの機能を無視しますので、マニュアルで確認してください。

5 ：読み取り部のサイズを指定します。

6 ：読み取ったデータをそのまま表示するか、白黒を反転して表示するかを選択します。通常はノーマルです。

＊ ：スキャナ・スタート。スキャナによっては、本体側でもスタートスイッチを押さなければならないものもあります。なお、ESC キーにより途中でスキャナを停止させることができます。

H.CLR ：画面をクリアします（CLS 2)。

SPACE ：スペースキーを押すたびに、テキスト画面／グラフィック画面が交互に切り換わります。

ESC ：GMENU.COMによって起動した場合にはメニュー画面（写真 1）へ戻ります。また、MS-DOSのコマンドラインから起動した場合にはMS-DOSへ戻ります。

以上が、本書で用意したグラフック・ツールの全機能です。一度にすべての機能を覚えるのは大変かもしれませんが、ゲーム制作においてはどれ一つとして欠かすことができないものばかりです。ぜひ、うまく使いこなして市販ソフトを越えるゲームにチャレンジしてください。

付録　8086 ニモニック一覧表

ニモニックのオペランドで使われている記号の意味

オペランド	意　　味
reg	8または16ビットの汎用レジスタ
reg8	8ビットの汎用レジスタ
reg16	16ビットの汎用レジスタ
mem	8または16ビット·メモリロケーション
mem8	8ビット·メモリロケーション
mem16	16ビット·メモリロケーション
mem32	32ビット·メモリロケーション
acc	AX、ALレジスタ
sreg	セグメントレジスタ
imm	8ビットまたは16ビットの数値
imm8	8ビットの数値
imm16	16ビットの数値
nearproc	現在の命令が置かれているコードセグメント内のプロシージャ
farproc	別なコードセグメント内のプロシージャ
nlabel	命令が置かれているコードセグメント内のラベル
flabel	別なコードセグメント内のラベル
slabel	命令の終わりから-128～+127バイトの範囲のラベル
memptr16	制御が移されようとしているオフセットが格納してあるワード
memptr32	制御が移されようとしているオフセットとセグメントが格納してあるダブルワード
regptr16	制御が移されようとしているオフセット格納してあるレジスタ
pvalue	スタックからPOPされるバイト数(偶数)
exop	コプロセッサの命令中にエンコードされる数値(0～63)

８０８６のレジスタ紹介

名前	説　　明	
AX	アキュムレータ	(16ビット)
AL	AXの下位8ビット	(アキュムレータ8ビット)
AH	AXの上位8ビット	
BX	ベース·レジスタ	(16ビット)
BL	BXの下位8ビット	
BH	BXの上位8ビット	
CX	カウンタ·レジスタ	(16ビット)
CL	CXの下位8ビット	
CH	CXの上位8ビット	
DX	データ·レジスタ	(16ビット)
DL	DXの下位8ビット	
DH	DXの上位8ビット	
SI	ソース·インデックス·レジスタ	(16ビット)
DI	ディスティネーション·インデックス·ジスタ	(16ビット)
CS	コード·セグメント·レジスタ	(16ビット)
DS	データ·セグメント·レジスタ	(16ビット)
ES	エクストラ·セグメント·レジスタ	(16ビット)
SS	スタック·セグメント·レジスタ	(16ビット)
SP	スタック·ポインタ	(16ビット)
BP	ベース·ポインタ	(16ビット)
IP	インストラクション·ポインタ	(16ビット)
FL	フラグ·レジスタ	(16ビット)

フラグ記号の意味

·	変化なし
?	不定
X	結果に従って変化する
0	リセット
1	セット
r	退避した値をｽﾄｱする

フラグの名称

AF	補助キャリーフラグ
CF	キャリーフラグ
PF	パリティーフラグ
SF	サインフラグ
ZF	ゼロフラグ
DF	ディクションフラグ
IF	インターラプトフラグ
OF	オーバーフローフラグ
TF	トラップフラグ

クロック記号の意味

記号	意味
N	N回かけあわせる
/	A／B　AまたはB
～	A～B　AからB
+EA	ダイレクト16ビット・オフセット・アドレス……6 ベースまたはインデックス・レジスタによるインダイレクト……5 インデックス・レジスタとベース・レジスタとの和によるインダイレクト……7 or 8 ディスプレイスメントを伴ったベースまたはインデックス・レジスタによるインダイレクト……9 ディスプレイスメントを伴ったインデックス・レジスタとベース・レジスタの和によるインダイレクト……11 or 12 (注)奇数アドレスに対する16ビット・オペランドでは4クロック加える。 またセグメント・オーバーライドにはさらに2クロック加える

オペレーションコード・フィールド

名前	説明
W	ワード／バイト・フィールド　(0 or 1)
reg	レジスタ・フィールド　(000～111)
sreg	セグメント・レジスタ・フィールド　(00～11)
r/m	レジスタ／メモリ・フィールド　(000～111)
mod	モード・フィールド　(00～10)
S:W	S:W=01 のとき data=16ビット、それ以外はdata=8ビットS:W=11 のときバイトデータのサインが拡張されて16ビット・オペランドを作る
XXX	ESCオペレーション・コードの初めの3ビット
YYY	ESCオペレーション・コードの2番目の3ビット

8または16ビット汎用レジスタの選択

reg or r/m *	W=0	W=1
000	AL	AX
001	CL	CX
010	DL	DX
011	BL	BX
100	AH	SP
101	CH	BP
110	DH	SI
111	BH	DI

* r/mはmodのない場合

セグメント・レジスタの選択

sreg	内容
00	ES
01	CS
10	SS
11	DS

メモリ・アドレッシング

r/m ＼ mod	00	01	10
000	[BX+SI]	[BX+SI+disp8]	[BX+SI+disp16]
001	[BX+DI]	[BX+DI+disp8]	[BX+DI+disp16]
010	[BP+SI]	[BP+SI+disp8]	[BP+SI+disp16]
011	[BP+DI]	[BP+DI+disp8]	[BP+DI+disp16]
100	[SI]	[SI+disp8]	[SI+disp16]
101	[DI]	[DI+disp8]	[DI+disp16]
110	[DIRECT ADDRESS]	[BP+disp8]	[BP+disp16]
111	[BX]	[BX+disp8]	[BX+disp16]

8086ニモニック一覧表（機能別アルファベット順）

加算命令

ニモニック	オペランド	第1バイト HEX	第1バイト 7 6 5 4 3 2 1 0	第2バイト HEX	第2バイト 7 6 5 4 3 2 1 0	バイト数	クロック数	フラグ ODISZAPC	内容
AAA		37	0 0 1 1 0 1 1 1			1	4	?··??X?X	Ascii Adjust for Addition 10進アスキーコード間における加算結果をALレジスタに求めたとして、その補正を行う。桁上がりはAHに入る 【使用例】 MOV AH,00H :AH←00H MOV AL,35H :AL←'5' ADD AL,37H :AL←'5'+'7' AAA :AL＝02H (AH=01H)
ADC	reg,reg		0 0 0 1 0 0 0 W		1 1 reg r/m	2	3	X··XXXXX	ADd with Carry キャリーを含む加算を行う
	mem,reg		0 0 0 1 0 0 0 W		mod reg r/m	2～4	16+EA	X··XXXXX	
	reg,mem		0 0 0 1 0 0 1 W		mod reg r/m	2～4	9+EA	X··XXXXX	
	reg,imm		1 0 0 0 0 0 S W		1 1 0 1 0 r/m	3～4	4	X··XXXXX	
	mem,imm		1 0 0 0 0 0 S W		mod 0 1 0 r/m	3～6	17+EA	X··XXXXX	【使用例】
	acc,imm		0 0 0 1 0 1 0 W			2～3	4	X··XXXXX	ADC AX,BX :AX←AX+BX+CF
ADD	reg,reg		0 0 0 0 0 0 0 W		1 1 reg r/m	2	3	X··XXXXX	ADDition 加算命令
	mem,reg		0 0 0 0 0 0 0 W		mod reg r/m	2～4	16+EA	X··XXXXX	
	reg,mem		0 0 0 0 0 0 1 W		mod reg r/m	2～4	9+EA	X··XXXXX	
	reg,imm		1 0 0 0 0 0 S W		1 1 0 0 0 r/m	3～4	4	X··XXXXX	
	mem,imm		1 0 0 0 0 0 S W		mod 0 0 0 r/m	3～6	17+EA	X··XXXXX	【使用例】
	acc,imm		0 0 0 0 0 1 0 W			2～3	4	X··XXXXX	ADD AX,BX :AX←AX+BX
DAA		27	0 0 1 0 0 1 1 1			1	4	?··XXXXX	Decimal Adjust for Addition 2進化10進数における加算結果をALに求めたとして、その補正を行う 【使用例】 MOV AL,35H :AL←35H ADD AL,46H :AL＝7BH DAA :AL＝81H
INC	reg8	FE	1 1 1 1 1 1 1 0		1 1 0 0 0 r/m	2	3	X··XXXX·	INCrement by 1
	mem		1 1 1 1 1 1 1 W		mod 0 0 0 r/m	2～4	15+EA	X··XXXX·	オペランドの内容を+1する
	reg16		0 1 0 0 0 reg			1	2	X··XXXX·	【使用例】 INC AX

減算命令

ニモニック	オペランド	第１バイト HEX	第１バイト 7 6 5 4 3 2 1 0	第２バイト HEX	第２バイト 7 6 5 4 3 2 1 0	バイト数	クロック数	フラグ ODISZAPC	内容
AAS		3F	0 0 1 1 1 1 1 1			1	4	?··??X?X	Ascii Adjust for Subtraction 10進ｱｽｷｰｺｰﾄﾞ間の減算結果をALﾚｼﾞｽﾀに求めたとして、その補正を行う 【使用例】 MOV AH,00H :AH←00H MOV AL,35H :AL←'5' SUB AL,36H :AL←'5'-'6' AAS :AL＝09H (AH=FFH)
CMP	reg,reg mem,reg reg,mem reg,imm mem,imm acc,imm		0 0 1 1 1 0 0 W 0 0 1 1 1 0 0 W 0 0 1 1 1 0 1 W 1 0 0 0 0 0 S W 1 0 0 0 0 0 S W 0 0 1 1 1 1 0 W		1 1 reg r/m mod reg r/m mod reg r/m 1 1 1 1 1 r/m mod 1 1 1 r/m	2 2～4 2～4 3～4 3～6 2～3	3 9+EA 9+EA 4 10+EA 4	X··XXXXX X··XXXXX X··XXXXX X··XXXXX X··XXXXX X··XXXXX	CoMPare destination to source 比較 【使用例】 CMP AX,BX JNE ****
DAS		2F	0 0 1 0 1 1 1 1			1	4	?··XXXXX	Decimal Adjust for Subtraction 2進化10進数における減算結果をﾚｼﾞｽﾀALに求めたとしてその補正をする 【使用例】 MOV AL,63H :AL←63H SUB AL,35H :AL＝2EH DAS :AL＝28H
DEC	reg8 mem reg16	FE	1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 W 0 1 0 0 1 reg		1 1 0 0 1 r/m mod 0 0 1 r/m	2 2～4 1	3 15+EA 2	X··XXXX· X··XXXX· X··XXXX·	DECrement by 1 ｵﾍﾟﾗﾝﾄﾞの内容を-1する 【使用例】 DEC AX
NEG	reg mem		1 1 1 1 0 1 1 W 1 1 1 1 0 1 1 W		1 1 0 1 1 r/m mod 0 1 1 r/m	2 2～4	3 16+EA	X··XXXXX X··XXXXX	NEGate ﾚｼﾞｽﾀやﾒﾓﾘの内容の補数 【使用例】 NEG AX
SBB	reg,reg mem,reg reg,mem reg,imm mem,imm acc,imm		0 0 0 1 1 0 0 W 0 0 0 1 1 0 0 W 0 0 0 1 1 0 1 W 1 0 0 0 0 0 S W 1 0 0 0 0 0 S W 0 0 0 1 1 1 0 W		1 1 reg r/m mod reg r/m mod reg r/m 1 1 0 1 1 r/m mod 0 1 1 r/m	2 2～4 2～4 3～4 3～6 2～3	3 16+EA 9+EA 4 17+EA 4	X··XXXXX X··XXXXX X··XXXXX X··XXXXX X··XXXXX X··XXXXX	SuBtract with Borrow ｷｬﾘｰを含む減算をおこなう 【使用例】 SBB AX,BX :AX←AX-BX-CF
SUB	reg,reg mem,reg reg,mem reg,imm mem,imm acc,imm		0 0 1 0 1 0 0 W 0 0 1 0 1 0 0 W 0 0 1 0 1 0 1 W 1 0 0 0 0 0 S W 1 0 0 0 0 0 S W 0 0 1 0 1 1 0 W		1 1 reg r/m mod reg r/m mod reg r/m 1 1 1 0 1 r/m mod 1 0 1 r/m	2 2～4 2～4 3～4 3～6 2～3	3 16+EA 9+EA 4 17+EA 4	X··XXXXX X··XXXXX X··XXXXX X··XXXXX X··XXXXX X··XXXXX	SUBtraction 減算命令 【使用例】 SUB AX,BX :AX←AX-BX

乗算命令

ニモニック	オペランド	第1バイト HEX	第1バイト 7 6 5 4 3 2 1 0	第2バイト HEX	第2バイト 7 6 5 4 3 2 1 0	バイト数	クロック数	フラグ ODISZAPC	内容
AAM		D4	1 1 0 1 0 1 0 0	0A	0 0 0 0 1 0 1 0	2	83	?・・XX?X?	Ascii Adjust for Multiplication 10進アスキーコード間における乗算結果をALレジスタに求めたとして、その補正を行う。 乗算命令にはアスキーコード数どうしの命令は用意されていないので上位4ビットはあらかじめ0クリアしておかなければならない 【動作】 AH←ALを10で割った商 AL←ALを10で割った余り 【使用例】'3'×'5' MOV AL,33H :AL←'3' MOV BL,35H :BL←'5' AND AL,0FH :AL＝3 AND BL,0FH :BL＝5 MUL BL :AL＝0FH AAM :AX＝0105H OR AX,3030H :AX＝'15'
IMUL	reg8 mem8 reg16 mem16	F6 F6 F7 F7	1 1 1 1 0 1 1 0 1 1 1 1 0 1 1 0 1 1 1 1 0 1 1 1 1 1 1 1 0 1 1 1		1 1 1 0 1 r/m mod 1 0 1 r/m 1 1 1 0 1 r/m mod 1 0 1 r/m	2 2〜4 2 2〜4	80-98 (86-104)+EA 128-154 (134-160)+EA	X・・????X X・・????X X・・????X X・・????X	Integer MULtiplication 符号付きの乗算命令 1バイトどうし/2バイトどうしの乗算が可能。1バイト乗算の時には、ALレジスタとオペランド間で乗算が行われる。結果はAL、AHに入る。 2バイト乗算の時にはAXレジスタとオペランド間で乗算が行われる。 結果はAX、DXに入る 【使用例】 -2×3 MOV AL,0FEH :AL← -2 MOV BL,3 :BL← +3 IMUL BL :AX＝ -6
MUL	reg8 mem8 reg16 mem16	F6 F6 F7 F7	1 1 1 1 0 1 1 0 1 1 1 1 0 1 1 0 1 1 1 1 0 1 1 1 1 1 1 1 0 1 1 1		1 1 1 0 0 r/m mod 1 0 0 r/m 1 1 1 0 0 r/m mod 1 0 0 r/m	2 2〜4 2 2〜4	70-77 (76-83)+EA 118-133 (124-139)+EA	X・・????X X・・????X X・・????X X・・????X	MULtiplication unsigned 符号なしの乗算命令 1バイトどうし/2バイトどうしの乗算が可能。1バイト乗算の時にはALレジスタとオペランド間で乗算が行われる。結果はAL、AHに入る。 2バイト乗算の時には、AXレジスタとはオペランド間で乗算が行われる。結果はAX、DXに入る 【使用例】 2×3 MOV AL,2 :AL← 2 MOV BL,3 :BL← 3 MUL BL :AX＝ 6

除算命令

ニモニック	オペランド	第1バイト HEX	第1バイト 7 6 5 4 3 2 1 0	第2バイト HEX	第2バイト 7 6 5 4 3 2 1 0	バイト数	クロック数	フラグ ODISZAPC	内容
AAD		D5	1 1 0 1 0 1 0 1	0A	0 0 0 0 1 0 1 0	2	60	?‥XX?X?	Ascii Adjust for Division 除算を行う時にAXレジスタの補正をする。除算命令にはアスキーコード数どうしの命令は用意されていないので除算を行う前に演算に適した形に補正しなければならない。他の補正命令は演算後に補正をするが、この場合は演算前に補正する 【動作】 AL←AH×10＋AL　　AH←0 【使用例】'15'÷'7' MOV　AX,3135H　:AX←'15' MOV　BL,37H　:BL←'7' AND　AX,0F0FH　:AX＝0105H AND　BL,0FH　:BL＝07H AAD　:AX＝000FH DIV　BL　:AX＝0102H
CBW		98	1 0 0 1 1 0 0 0			1	2	‥‥‥‥	Convert Byte to Word ALレジスタの符号をAHレジスタに拡張する 【動作】 ALレジスタの最上位ビットが0ならAHレジスタを00Hに、1ならAHレジスタをFFHにする
CWD		99	1 0 0 1 1 0 0 1			1	5	‥‥‥‥	Convert Word to Doubleword AXレジスタの符号をDXレジスタに拡張する 【動作】 AXレジスタの最上位ビットが0ならDXレジスタを0000Hに、1ならDXレジスタをFFFFHにする
IDIV	reg8 mem8 reg16 mem16	F6 F6 F7 F7	1 1 1 1 0 1 1 0 1 1 1 1 0 1 1 0 1 1 1 1 0 1 1 1 1 1 1 1 0 1 1 1		1 1 1 1 1 r/m mod 1 1 1 r/m 1 1 1 1 1 r/m mod 1 1 1 r/m	2 2～4 2 2～4	101-112 (107-118)+EA 165-184 (171-190)+EA	?‥????? ?‥????? ?‥????? ?‥?????	Integer DIVision 符号付き除算命令 8ビットの場合は、AXレジスタが割られる数で、商＝ALレジスタ、余り＝AHレジスタとなる。 16ビットの場合は、DXレジスタに割られる数の上位、AXレジスタに下位を格納する。結果は、商＝AXレジスタ、余り＝DXレジスタとなる 【使用例】3÷(-2) MOV　AX,3　:AX←3 MOV　BL,0FEH　:BL←-2 IDIV　BL(AX=01FFH)　:AX÷BL
DIV	reg8 mem8 reg16 mem16	F6 F6 F7 F7	1 1 1 1 0 1 1 0 1 1 1 1 0 1 1 0 1 1 1 1 0 1 1 1 1 1 1 1 0 1 1 1		1 1 1 1 0 r/m mod 1 1 0 r/m 1 1 1 1 0 r/m mod 1 1 0 r/m	2 2～4 2 2～4	80-90 (86-96)+EA 144-162 (150-168)+EA	?‥????? ?‥????? ?‥????? ?‥?????	DIVision unsigned 符号なし除算命令 8ビットの場合は、AXレジスタが割られる数で、商＝ALレジスタ、余り＝AHレジスタとなる。 16ビットの場合は、DXレジスタに割られる数の上位、AXレジスタに下位を格納する。結果は、商＝AXレジスタ、余り＝DXレジスタとなる 【使用例】3÷2 MOV　AX,3　:AX←3 MOV　BL,2　:BL←2 DIV　BL(AX=0101H)　:AX÷BL

データ転送命令

ニモニック	オペランド	第1バイト HEX	第1バイト 7 6 5 4 3 2 1 0	第2バイト HEX	第2バイト 7 6 5 4 3 2 1 0	バイト数	クロック数	フラグ ODISZAPC	内容
IN	acc,imm8		1 1 1 0 0 1 0 W			2	10	・・・・・・・・	INput ALまたはAXレジスタに、I/Oポートからデータを入力する
	acc,DX		1 1 1 0 1 1 0 W			1	8	・・・・・・・・	
LAHF		9F	1 0 0 1 1 1 1 1			1	4	・・・・・・・・	Load AH from Flags フラグレジスタの下位8ビットをAHレジスタに代入する。また、AHにフラグデータを取り込めば、フラグデータを一般的なビットデータとして扱うことが可能。そのデータをSAHF命令でフラグレジスタに返すこともできる
LDS	reg16,mem32	C5	1 1 0 0 0 1 0 1		mod reg r/m	2〜4	16+EA	・・・・・・・・	Load pointer using DS 指定したレジスタとDSレジスタに第2オペランドで指定したアドレスから順にデータを取り込む
LEA	reg16,mem16	8D	1 0 0 0 1 1 0 1		mod reg r/m	2〜4	2+EA	・・・・・・・・	Load Effective Addres メモリオペランドによって指定されるオフセット値を、指定したレジスタに代入する
LES	reg16,mem32	C4	1 1 0 0 0 1 0 0		mod reg r/m	2〜4	16+EA	・・・・・・・・	Load pointer using ES 指定したレジスタとESレジスタに第2オペランドで指定したアドレスから順にデータを取り込む
MOV	reg,reg		1 0 0 0 1 0 0 W		1 1 reg r/m	2	2	・・・・・・・・	MOVe 第2オペランドのデータを第1オペランドへ代代入する
	mem,reg		1 0 0 0 1 0 0 W		mod reg r/m	2〜4	9+EA	・・・・・・・・	
	reg,mem		1 0 0 0 1 0 1 W		mod reg r/m	2〜4	8+EA	・・・・・・・・	
	mem,imm		1 1 0 0 0 1 1 W		mod 0 0 0 r/m	3〜6	10+EA	・・・・・・・・	
	reg,imm		1 0 1 1 W reg			2〜3	4	・・・・・・・・	
	acc,mem		1 0 1 0 0 0 0 W			3	10	・・・・・・・・	
	mem,acc		1 0 1 0 0 0 1 W			3	10	・・・・・・・・	
	sreg,reg16	8E	1 0 0 0 1 1 1 0		1 1 0 sreg r/m	2	2	・・・・・・・・	
	sreg,mem	8E	1 0 0 0 1 1 1 0		mod 0 sreg r/m	2〜4	8+EA	・・・・・・・・	
	reg16,sreg	8C	1 0 0 0 1 1 0 0		1 1 0 sreg r/m	2	2	・・・・・・・・	
	mem,sreg	8C	1 0 0 0 1 1 0 0		mod 0 sreg r/m	2〜4	9+EA	・・・・・・・・	
OUT	imm8,acc		1 1 1 0 0 1 1 W			2	10	・・・・・・・・	OUTput I/Oポートに、ALまたはAXレジスタの値を出力する
	DX,acc		1 1 1 0 1 1 1 W			1	8	・・・・・・・・	

（「データ転送命令」続く）

ニモニック	オペランド	第1バイト		第2バイト		バイト数	クロック数	フラグ	内容
		HEX	7 6 5 4 3 2 1 0	HEX	7 6 5 4 3 2 1 0			ODISZAPC	
POP	mem reg sreg	8F	1 0 0 0 1 1 1 1 0 1 0 1 1 reg 0 0 0 sreg1 1 1		mod 0 0 0 r/m	2～4 1 1	17+EA 8 8	········ ········ ········	POP word off stack スタックエリアの先頭からデータを取り出しオペランドへ返す
POPF		9D	1 0 0 1 1 1 0 1			1	8	rrrrrrrr	POP Flags off stack スタックエリアの先頭からフラグレジスタへデータを取り込む
PUSH	mem reg sreg	FF	1 1 1 1 1 1 1 1 0 1 0 1 0 reg 0 0 0 sreg1 1 0		mod 1 1 0 r/m	2～4 1 1	16+EA 10 10	········ ········ ········	PUSH word onto stack スタックエリアの先頭にオペランの内容を書き込む
PUSHF		9C	1 0 0 1 1 1 0 0			1	10	········	PUSH Flags onto stack スタックエリアの先頭にフラグレジスタの内容を書き込む
SAHF		9E	1 0 0 1 1 1 1 0			1	4	···XXXXX	Store AH into Flags フラグレジスタの下位8ビットにAHレジスタの値を代入する
XCHG	reg,reg mem,reg AX,reg16		1 0 0 0 0 1 1 W 1 0 0 0 0 1 1 W 1 0 0 1 0 reg		1 1 reg r/m mod reg r/m	2 2～4 1	4 17+EA 3	········ ········ ········	eXCHanGe 第1オペランドと第2オペランドの内容を交換する

XLAT		D7	1 1 0 1 0 1 1 1			1	11	········	Translate
									BXレジスタにALレジスタの内容を加算し、この値をオフセット値とし、セグメントをDSレジスタの値で参照されるアドレスからデータを取り出しALレジスタに格納する
									【使用例】
									MOV BX,1000H
									MOV [BX],2010H
									MOV AL,1 :AL←1
									XALT :AL←20H

論理演算命令

ニモニック	オペランド	第1バイト HEX	第1バイト 7 6 5 4 3 2 1 0	第2バイト HEX	第2バイト 7 6 5 4 3 2 1 0	バイト数	クロック数	フラグ ODISZAPC	内容
AND	reg,reg mem,reg reg,mem reg,imm mem,imm acc,imm		0 0 1 0 0 0 0 W 0 0 1 0 0 0 0 W 0 0 1 0 0 0 1 W 1 0 0 0 0 0 0 W 1 0 0 0 0 0 0 W 0 0 1 0 0 1 0 W		1 1 reg r/m mod reg r/m mod reg r/m 1 1 1 0 0 r/m mod 1 0 0 r/m	2 2～4 2～4 3～4 3～6 2～3	3 16+EA 9+EA 4 17+EA 4	0··XX?X0 0··XX?X0 0··XX?X0 0··XX?X0 0··XX?X0 0··XX?X0	logical AND 第1オペランドと第2オペランドとのANDを取る。結果は第1オペランドに入る
NOT	reg mem		1 1 1 1 0 1 1 W 1 1 1 1 0 1 1 W		1 1 0 1 0 r/m mod 0 1 0 r/m	2 2～4	3 16+EA	········ ········	logical NOT オペランドの各ビットの反転を行う
OR	reg,reg mem,reg reg,mem reg,imm mem,imm acc,imm		0 0 0 0 1 0 0 W 0 0 0 0 1 0 0 W 0 0 0 0 1 0 1 W 1 0 0 0 0 0 0 W 1 0 0 0 0 0 0 W 0 0 0 0 1 1 0 W		1 1 reg r/m mod reg r/m mod reg r/m 1 1 0 0 1 r/m mod 0 0 1 r/m	2 2～4 2～4 3～4 3～6 2～3	3 16+EA 9+EA 4 17+EA 4	0··XX?X0 0··XX?X0 0··XX?X0 0··XX?X0 0··XX?X0 0··XX?X0	logical OR 第1オペランドと第2オペランドとのORを取る。結果は第1オペランドに入る
RCL	reg,1 mem,1 reg,CL mem,CL		1 1 0 1 0 0 0 W 1 1 0 1 0 0 0 W 1 1 0 1 0 0 1 W 1 1 0 1 0 0 1 W		1 1 0 1 0 r/m mod 0 1 0 r/m 1 1 0 1 0 r/m mod 0 1 0 r/m	2 2～4 2 2～4	2 15+EA 8+4N 20+EA+4N	X······X X······X ?······X ?······X	Rotate through Carry Left キャリーと共にメモリまたはレジスタのビットを左へ回転させる。回転する回数は1またはレジスタで指定する C ← [←]
RCR	reg,1 mem,1 reg,CL mem,CL		1 1 0 1 0 0 0 W 1 1 0 1 0 0 0 W 1 1 0 1 0 0 1 W 1 1 0 1 0 0 1 W		1 1 0 1 1 r/m mod 0 1 1 r/m 1 1 0 1 1 r/m mod 0 1 1 r/m	2 2～4 2 2～4	2 15+EA 8+4N 20+EA+4N	X······X X······X ?······X ?······X	Rotate through Carry Right キャリーと共にメモリまたはレジスタのビットを右へ回転させる。回転る回数は1またはCLレジスタで指定する C → [→]
ROL	reg,1 mem,1 reg,CL mem,CL		1 1 0 1 0 0 0 W 1 1 0 1 0 0 0 W 1 1 0 1 0 0 1 W 1 1 0 1 0 0 1 W		1 1 0 0 0 r/m mod 0 0 0 r/m 1 1 0 0 0 r/m mod 0 0 0 r/m	2 2～4 2 2～4	2 15+EA 8+4N 20+EA+4N	X······X X······X ?······X ?······X	ROtate Left オペランドのビットを左へ回転させる。回転する回数は1またはCLレジスタで指定する C ← [←]

(「論理演算命令」続く)

ニモニック	オペランド	第1バイト HEX	第1バイト 7 6 5 4 3 2 1 0	第2バイト HEX	第2バイト 7 6 5 4 3 2 1 0	バイト数	クロック数	フラグ ODISZAPC	内容
ROR	reg,1 mem,1 reg,CL mem,CL		1 1 0 1 0 0 0 W 1 1 0 1 0 0 0 W 1 1 0 1 0 0 1 W 1 1 0 1 0 0 1 W		1 1 0 0 1 r/m mod 0 0 1 r/m 1 1 0 0 1 r/m mod 0 0 1 r/m	2 2～4 2 2～4	2 15+EA 8+4N 20+EA+4N	X······X X······X ?······X ?······X	ROtate Right オペランドのビットを右へ回転させる。回転する回数は1またはCLレジスタで指定する → C
SAL SHL	reg,1 mem,1 reg,CL mem,CL		1 1 0 1 0 0 0 W 1 1 0 1 0 0 0 W 1 1 0 1 0 0 1 W 1 1 0 1 0 0 1 W		1 1 1 0 0 r/m mod 1 0 0 r/m 1 1 1 0 0 r/m mod 1 0 0 r/m	2 2～4 2 2～4	2 15+EA 8+4N 20+EA+4N	X······X X······X ?······X ?······X	Shift Arithmetic Left SHift logical Left オペランドのビットを左へシフトする。シフトする数は1またはCLレジスタで指定する C ← ← 0
SAR	reg,1 mem,1 reg,CL mem,CL		1 1 0 1 0 0 0 W 1 1 0 1 0 0 0 W 1 1 0 1 0 0 1 W 1 1 0 1 0 0 1 W		1 1 1 1 1 r/m mod 1 1 1 r/m 1 1 1 1 1 r/m mod 1 1 1 r/m	2 2～4 2 2～4	2 15+EA 8+4N 20+EA+4N	X······X X······X ?······X ?······X	Shift Arithmetic Right オペランドのビットを右へ算術シフトする。シフトする数は1またはCLレジスタで指定する 符号→ → C
SHR	reg,1 mem,1 reg,CL mem,CL		1 1 0 1 0 0 0 W 1 1 0 1 0 0 0 W 1 1 0 1 0 0 1 W 1 1 0 1 0 0 1 W		1 1 1 0 1 r/m mod 1 0 1 r/m 1 1 1 0 1 r/m mod 1 0 1 r/m	2 2～4 2 2～4	2 15+EA 8+4N 20+EA+4N	X······X X······X ?······X ?······X	SHift logical Right オペランドのビットを右へシフトする。シフトする数は1またはCLレジスタで指定する 0 → → C
TEST	reg,reg mem,reg reg,imm mem,imm acc,imm		1 0 0 0 0 1 0 W 1 0 0 0 0 1 0 W 1 1 1 1 0 1 1 W 1 1 1 1 0 1 1 W 1 0 1 0 1 0 0 W		1 1 reg r/m mod reg r/m 1 1 0 0 0 r/m mod 0 0 0 r/m	2 2～4 3～4 3～6 2～3	3 9+EA 5 11+EA 4	0··XX?X0 0··XX?X0 0··XX?X0 0··XX?X0 0··XX?X0	TEST 第1オペランドと第2オペランドとのANDを取った結果をフラグに反映する。オペランドの値は変化しない
XOR	reg,reg mem,reg reg,mem reg,imm mem,imm acc,imm		0 0 1 1 0 0 0 W 0 0 1 1 0 0 0 W 0 0 1 1 0 0 1 W 1 0 0 0 0 0 0 W 1 0 0 0 0 0 0 W 0 0 1 1 0 1 0 W		1 1 reg r/m mod reg r/m mod reg r/m 1 1 1 1 0 r/m mod 1 1 0 r/m	2 2～4 2～4 3～4 3～6 2～3	3 16+EA 9+EA 4 17+EA 4	0··XX?X0 0··XX?X0 0··XX?X0 0··XX?X0 0··XX?X0 0··XX?X0	logcial eXclusive OR 第1オペランドと第2オペランドとのXORを取る。結果は第1オペランドに入る

分岐命令

ニモニック	オペランド	第1バイト HEX	第1バイト 7 6 5 4 3 2 1 0	第2バイト HEX	第2バイト 7 6 5 4 3 2 1 0	バイト数	クロック数	フラグ ODISZAPC	内容
CALL	nearproc regptr16 memptr16 farproc memptr32	E8 FF FF 9A FF	1 1 1 0 1 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 1 0 1 0 1 1 1 1 1 1 1 1		 1 1 0 1 0 r/m mod 0 1 0 r/m mod 0 1 1 r/m	3 2 2～4 5 2～4	19 16 21+EA 28 37+EA	········ ········ ········ ········ ········	CALL a procedure オペランドで示されるプロシージャをコールする
INT	3 imm8(≠3)	CC CD	1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 1			1 2	52 51	··0····· ··0·····	INTerrupt 内部割り込み処理の実行。 セグメント0の0～3FFHに、ベクタ・テーブルが用意されており、ジャンプ先を番号によって指定する
INTO		CE	1 1 0 0 1 1 1 0			1	53/4	··X·····	INTerrupt if Overflow オーバーフローフラグが1ならタイプ4の割り込みを発生させる
IRET		CF	1 1 0 0 1 1 1 1			1	24	rrrrrrrr	Interrupt RETurn 割り込み処理からの復帰
JA JNBE	slabel	77	0 1 1 1 0 1 1 1			2	16/4	········	Jump if Above Jump if Not Below nor Equal より大ならジャンプする 【動作】 CF=0 AND ZF=0でジャンプ
JAE JNB	slabel	73	0 1 1 1 0 0 1 1			2	16/4	········	Jump if Above or Equal Jump if Not Below より大か等しければジャンプする 【動作】 CF=0 でジャンプ
JB JNAE	slabel	72	0 1 1 1 0 0 1 0			2	16/4	········	Jump if Below Jump if Not Above nor Equal より小ならジャンプする 【動作】 CF=1 でジャンプ
JBE JNA	slabel	76	0 1 1 1 0 1 1 0			2	16/4	········	Jump if Below or Equal Jump if Not Above より小か等しければジャンプする 【動作】 CF=1 OR ZF=1 でジャンプ
JCXZ	slabel	E3	1 1 1 0 0 0 1 1			2	18/6	········	Jump if CX is Zero CXレジスタ=0の場合にジャンプする
JE JZ	slabel	74	0 1 1 1 0 1 0 0			2	16/4	········	Jump if Equal Jump if Zero 等しければジャンプする 【動作】 ZF=1 でジャンプ

(「分岐命令」続く)

ニモニック	オペランド	第１バイト		第２バイト		バイト数	クロック数	フラグ	内　　　容
		HEX	7 6 5 4 3 2 1 0	HEX	7 6 5 4 3 2 1 0			ODISZAPC	
JG JNLE	slabel	7F	0 1 1 1 1 1 1 1			2	16/4	·········	Jump if Greater Jump if Not Less nor Equal より大であればジャンプする 【動作】 ZF=0 AND SF=OFでジャンプ
JGE JNL	slabel	7D	0 1 1 1 1 1 0 1			2	16/4	·········	Jump if Greater or Equal Jump if Not Less 以上であればジャンプする 【動作】 SF=OF でジャンプ
JL JNGE	slabel	7C	0 1 1 1 1 1 0 0			2	16/4	·········	Jump if Less Jump if Not Greater nor Equal より小であればジャンプする 【動作】 SF≠OF でジャンプ
JLE JNG	slabel	7E	0 1 1 1 1 1 1 0			2	16/4	·········	Jump if Less or Equal Jump if Not Greater 以下であればジャンプする 【動作】 ZF=1 OR SF≠OFでジャンプ
JMP	nlabel slabel regptr16 memptr16 flabel memptr32	E9 EB FF FF EA FF	1 1 1 0 1 0 0 1 1 1 1 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 0 1 0 1 1 1 1 1 1 1 1		 1 1 1 0 0 r/m mod 1 0 0 r/m mod 1 0 1 r/m	3 2 2 2～4 5 2～4	15 15 11 18+EA 15 24+EA	········· ········· ········· ········· ········· ·········	JuMP オペランドで示された場所にジャンプする
JNE JNZ	slabel	75	0 1 1 1 0 1 0 1			2	16/4	·········	Jump if Not Equal Jump if Not Zero 等しくなければジャンプする 【動作】 ZF=0 でジャンプ
JNO	slabel	71	0 1 1 1 0 0 0 1			2	16/4	·········	Jump if Not Overflow オーバーフローでなければジャンプ 【動作】 OF=0 でジャンプ
JNP JPO	slabel	7B	0 1 1 1 1 0 1 1			2	16/4	·········	Jump if Not Parity Jump if Parity Odd パリティが奇数ならジャンプ 【動作】 PF=0 でジャンプ
JNS	slabel	79	0 1 1 1 1 0 0 1			2	16/4	·········	Jump if Not Sign サインフラグが0ならジャンプする 【動作】 SF=0 でジャンプ
JO	slabel	70	0 1 1 1 0 0 0 0			2	16/4	·········	Jump if Overflow オーバーフローであればジャンプ 【動作】 OF=1 でジャンプ

ニモニック	オペランド	第1バイト		第2バイト		バイト数	クロック数	フラグ	内容
		HEX	7 6 5 4 3 2 1 0	HEX	7 6 5 4 3 2 1 0			ODISZAPC	
JP JPE	slabel	7A	0 1 1 1 1 0 1 0			2	16/4	········	Jump if Parity Jump if Parity Even パリティが偶数ならジャンプ 【動作】 PF=1 でジャンプ
JS	slabel	78	0 1 1 1 1 0 0 0			2	16/4	········	Jump if Sign サインフラグが1ならジャンプする 【動作】 SF=1 でジャンプ

LOOP	slabel	E2	1 1 1 0 0 0 1 0			2	17/5	········	LOOP
									CXレジスタから1を引いてCXレジスタが0でなければループする
									（フラグには影響を与えない）
									【動作】CX＝CX-1
									CX≠0であればジャンプする
									【使用例】 10から1までの和
									MOV BX,0
									MOV CX,0AH
									LABEL1: ADD BX,CX
									LOOP LABEL1

LOOPE
LOOPZ
	slabel
	slabel
		E1	1 1 1 0 0 0 0 1			2	18/6	········	LOOP if Equal
									LOOP if Zero
									CXレジスタから1を引いてCXレジスタが0でなく、ゼロフラグが1ならばループする
									（フラグには影響を与えない）
									【動作】
									CX=CX-1 ZF=1かつCX≠0 でループ
									【使用例】
									ここではDS:100H～DS:1FFHまでのﾒﾓﾘの内容で0でないものを見つけたら抜け出すような例を示す
									MOV DI,0FFH
									MOV CX,100H
									LABEL1: INC DI
									CMP BYTE PTR [DI],0
									LOOPE LABEL1

（「分岐命令」続く）

ニモニック	オペランド	第1バイト HEX	第1バイト 7 6 5 4 3 2 1 0	第2バイト HEX	第2バイト 7 6 5 4 3 2 1 0	バイト数	クロック数	フラグ ODISZAPC	内容
LOOPNE LOOPNZ	slabel slabel	E0	1 1 1 0 0 0 0 0			2	19/5	·········	LOOP if Not Equal LOOP if Not Zero CXレジスタから1を引いてCXレジスタが0でなく、ゼロフラグが0ならばループする （フラグには影響を与えない） 【動作】 CX=CX-1　ZF=0かつCX≠0 でループ 【使用例】 ここでは DS:100H～DS:1FFHまでのメモリの内容で0を見つけたら抜け出すような例を示す 　　MOV　DI,0FFH 　　MOV　CX,100H LABEL1:　INC　DI 　　CMP　BYTE PTR [DI],0 　　LOOPNE　LABEL1
RET(near) (far)		C3 CB	1 1 0 0 0 0 1 1 1 1 0 0 1 0 1 1			1 1	8 18	········· ·········	RETurn from procedure プロシージャからの復帰
RET(near) (far)	pvalue （偶数）	C2 CA	1 1 0 0 0 0 1 0 1 1 0 0 1 0 1 0			3 3	12 17	········· ·········	RETurn from procedure プロシージャから復帰し、スタックポインタに pvalue 値を加算する

ストリング命令

ニモニック	オペランド	第1バイト HEX	第1バイト 7 6 5 4 3 2 1 0	第2バイト HEX	第2バイト 7 6 5 4 3 2 1 0	バイト数	クロック数	フラグ ODISZAPC	内容
CMPSB CMPSW		A6 A7	1 0 1 0 0 1 1 0 1 0 1 0 0 1 1 1			1 1	22 or 9+22N 22 or 9+22N	X··XXXXX X··XXXXX	CoMPare String for Byte CoMPare String for Word セグメントDSレジスタ、オフセットSIレジスタで示されるメモリの内容とセグメントESレジスタ、オフセットDIレジスタで示されるメモリの内容をバイト（ワード）単位で比較する。SI、DIレジスタ値は、この命令の実行後DF=0であれば+1(+2)、DF=1であれば-1(-2)だけ更新される。 REPZ/REPNZなどと組み合わせてブロック比較が可能。この時の繰り返し数はCXレジスタの値が使われる。 CXレジスタは自動的に更新されCX=0かREPZやREPNZでの条件が成立しなければ実行終了となる

ニモニック	オペランド	第1バイト		第2バイト		バイト数	クロック数	フラグ	内容
		HEX	7 6 5 4 3 2 1 0	HEX	7 6 5 4 3 2 1 0			ODISZAPC	
LODSB LODSW		AC AD	1 0 1 0 1 1 0 0 1 0 1 0 1 1 0 1			1 1	12 or 9+13N 12 or 9+13N	········ ········	LOaD String for Byte LOaD String for Word セグメントDSレジスタ、オフセットSIレジスタで示されるメモリの内容を、バイト単位であればALレジスタに、ワード(2バイト)単位であればAXレジスタにロードする。SIレジスタは、この命令の実行後DF=0であれば+1(+2)、DF=1であれば-1(-2)だけ更新される。REP命令と組み合わせて連続動作が可能。この時の繰り返し数はCXレジスタの値が使われる。 CXレジスタは自動的に更新されCX=0になれば実行終了となる
MOVSB MOVSW		A4 A5	1 0 1 0 0 1 0 0 1 0 1 0 0 1 0 1			1 1	18 or 9+17N 18 or 9+17N	········ ········	MOVe String for Byte MOVe String for Word セグメントDSレジスタ、オフセットSIレジスタで示されるメモリの内容をセグメントESレジスタ、オフセットDIレジスタで示されるメモリの内容へバイト（ワード）単位で転送する。SI、DIレジスタの値はこの命令の実行後DF=0であれば+1(+2)、DF=1であれば-1(-2)だけ更新される。 REP命令と組み合わせて連続動作が可能。この時の繰り返し数はCXレジスタの値が使われる。 CXレジスタは自動的に更新されCX=0になれば実行終了となる
SCASB SCASW		AE AF	1 0 1 0 1 1 1 0 1 0 1 0 1 1 1 1			1 1	15 or 9+15N 15 or 9+15N	X··XXXXX X··XXXXX	SCAn String for Byte SCAn String for Word セグメントESレジスタ、オフセトDIレジスタで示されるメモリの内容と、バイト単位であればALレジスタ、ワード(2バイト)単位であればAXレジスタの値との比較を行う。 DIレジスタの値はこの命令の実行後DF=0であれば+1(+2)、DF=1であれば-1(-2)だけ更新される。REPZ/REPNZなどと組み合わせて、連続比較が可能。この時の繰り返し数はCXレジスタの値が使われる。CXレジスタは自動的に更新され、CX=0かREPZやREPNZでの条件が成立しなければ実行終了となる

(「ストリング命令」続く)

ニモニック	オペランド	第1バイト		第2バイト		バイト数	クロック数	フラグ	内容
		HEX	7 6 5 4 3 2 1 0	HEX	7 6 5 4 3 2 1 0			ODISZAPC	
STOSB STOSW		AA AB	1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 1			1 1	11 or 9+10N 11 or 9+10N	········ ········	STOre String for Byte STOre String for Word セグメントESレジスタ、オフセットDIレジスタで示されるメモリの内容をバイト単位であればALレジスタ、ワード(2バイト)単位であればAXレジスタの内容に書き換える。 DIレジスタの値はこの命令の実行後DF=0であれば+1(+2)、DF=1であれば-1(-2)だけ更新される。REP命令と組み合わせて連続動作が可能。繰り返し数はCXレジスタの値が使われる。 CXレジスタは自動的に更新され、CX=0になれば実行が終了となる

ストリング・プリフィックス命令

ニモニック	オペランド	第1バイト		第2バイト		バイト数	クロック数	フラグ	内容
		HEX	7 6 5 4 3 2 1 0	HEX	7 6 5 4 3 2 1 0			ODISZAPC	
REP REPE REPZ		F3	1 1 1 1 0 0 1 1			1	2	········ ········ ········	Repeat while CX≠0 Repeat while CX≠0 & ZF=1 Repeat while CX≠0 & ZF=1 ストリング命令の前に置き、次のストリング命令をCX≠0かつZF=1となっている間、実行する。 REP/REPE/REPZはいずれも同じマシン語コードであり、1回の指定のみ有効となっている
REPNE REPNZ		F2	1 1 1 1 0 0 1 0			1	2	········ ········	Repeat while CX≠0 & ZF=0 Repeat while CX≠0 & ZF=0 ストリング命令の前に置き、次のストリング命令をCX≠0かつZF=0となっている間、実行する。 REPNE/REPNZはいずれも同じマシン語コードであり、1回の指定のみ有効となっている

フラグ制御命令

ニモニック	オペランド	第1バイト		第2バイト		バイト数	クロック数	フラグ	内容
		HEX	7 6 5 4 3 2 1 0	HEX	7 6 5 4 3 2 1 0			ODISZAPC	
CLC		F8	1 1 1 1 1 0 0 0			1	2	・・・・・・・0	CLear Carry flag キャリーフラグをクリア CF=0
CLD		FC	1 1 1 1 1 1 0 0			1	2	・0・・・・・・	CLear Direction flag ディレクションフラグをクリア DF=0
CLI		FA	1 1 1 1 1 0 1 0			1	2	・・0・・・・・	CLear Interrupt flag 割り込みフラグをクリア IF=0
CMC		F5	1 1 1 1 0 1 0 1			1	2	・・・・・・・X	CoMplement Carry flag キャリーフラグの反転
STC		F9	1 1 1 1 1 0 0 1			1	2	・・・・・・・1	SeT Carry flag キャリーフラグをセット CF=1
STD		FD	1 1 1 1 1 1 0 1			1	2	・1・・・・・・	SeT Direction flag ディレクションフラグをセット DF=1
STI		FB	1 1 1 1 1 0 1 1			1	2	・・1・・・・・	SeT Interrupt flag 割り込みフラグをセット IF=1

CPU制御命令

ニモニック	オペランド	第1バイト		第2バイト		バイト数	クロック数	フラグ	内容
		HEX	7 6 5 4 3 2 1 0	HEX	7 6 5 4 3 2 1 0			ODISZAPC	
ESC	exop,reg exop,mem		1 1 0 1 1 X X X 1 1 0 1 1 X X X		1 1 Y Y Y r/m mod Y Y Y r/m	2 2～4	2 8+EA	・・・・・・・・ ・・・・・・・・	ESCape データバスにオペランドで指定したメモリの内容を設定する
HLT		F4	1 1 1 1 0 1 0 0			1	2	・・・・・・・・	HaLT CPUの実行停止
LOCK		F0	1 1 1 1 0 0 0 0			1	2	・・・・・・・・	LOCK bus バスのロック信号を設定
NOP		90	1 0 0 1 0 0 0 0			1	3	・・・・・・・・	No OPeration 何も実行しない
WAIT		9B	1 0 0 1 1 0 1 1			1	3+5N	・・・・・・・・	Wait testピンの信号がアクティブになるまで待つ

〈著者紹介〉

日高　徹（ひだか　とおる）
1949年　栃木県宇都宮市生まれ
早稲田大学商学部卒業後、商社やカー用品メーカーに十数年勤務。現在はフリーのゲームデザイナー。
市販された作品に『ホーンテッドケーブ』『マジックガーデン』『北斗の拳』『ガンダーラ』など。著書は『マシン語ゲームプログラミング』『PC-8801シリーズ　マシン語サウンド・プログラミング』（以上アスキー）『PC-8800シリーズ　はじめてのマシン語』『PC-9800シリーズ　はじめてのマシン語』『Z80　マシン語秘伝の書』（以上啓学出版）。
趣味はトレーニング。剣道三段。運転免許証を全種類持つ、隠れプロドライバーでもある。

青山　学（あおやま　まなぶ）
1958年　東京生まれ。
青山学院大学理工学部卒業。コンピュータエンジニアを経て、現在はゲームデザイナー。
大型コンピュータからパソコンまで、言語、機種にこだわらないのを信条としている。著書として『PC-8801シリーズ マシン語ゲームプログラミング』（アスキー）、『PC-9800シリーズ　はじめてのマシン語』、『8086マシン語秘伝の書』（以上啓学出版）がある。
趣味は、スキー、テニスなど。

POPCOM　BOOKS
PC-9801シリーズ　マシン語ゲームグラフィックス

1991年4月10日　　初版第1刷発行
著　者　日高　徹・青山　学
発行人　相賀徹夫
発行所　小　学　館
〒101-01　東京都千代田区一ツ橋2-3-1
☎業務(03)　3230-5333
☎販売(03)　3230-5748　振替 東京8-200
編　集　株式会社　新企画社
〒101　東京都千代田区神田神保町3-3-7
昭和第2ビル　☎(03)　3263-6940
印　刷　大日本印刷株式会社

ISBN4 - 09 - 385082 - 8

ISBN4-09-385082-8 C3055 P4300E

定価4,300円(本体4,175円)